REAL-TIME EDGE ANALYTIC USING LSTM-TRACKING

by

Shrey Mohan

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2019

Approved by:

Dr. Hamed Tabkhi

Dr. Chen Chen

Dr. Omid Shoghli

ABSTRACT

SHREY MOHAN. Real-time edge analytic using lstm-tracking . (Under the direction of DR. HAMED TABKHI)

Tracking objects or specifically pedestrians implies that we correctly detect and re-identify(re-id) them throughout the video stream. To accomplish this we need to run these algorithms on every frame of the video which is difficult in real-time as these networks are compute intensive. To make the system near real-time we use smaller detection and re-id networks, namely OpenPose at a lower network resolution and MobileNet-v2 for feature extraction and matching respectively. This end-to-end pedestrian detection and re-id pipeline run efficiently on embedded platforms, with a trade-off in accuracy. The reason for this decrease in accuracy mainly comes from the detection algorithm which is not running in its full potential due to memory and power constraints of the edge device. Also, in scenarios like occlusions specifically dynamic occlusion where pedestrians cross each other, the re-id network fails again due to the missed detection. To deal with these limitations we explored algorithms which can understand movement patterns of different pedestrians and predict their future positions. By knowing their future positions we do not solely rely on the detection network and can replace any miss-detection with the future prediction for that pedestrian. Similarly when a pedestrian is partially or fully occluded by another pedestrian and cannot be detected in the scene, we again use these future predictions for that pedestrian. In this way we envision to deal with scenarios like miss-detection incurred by the detection algorithm and occlusions which is very frequent in real world cases.

Long Short Term Memory (LSTM) neural networks have been proven to achieve state of the art performance for pattern recognition problems. They inherently have a memory cell which keeps track of all the relevant data they have seen and learn to

recognize the hidden patterns in it. We leveraged these pattern learning capabilities of LSTM in this research and trained it to predict future positions of the pedestrians in the scene. The LSTM is trained at a coarse-grain granularity of 5 frames per second using sequences shot at 60 frames per second. We then quantify its performance and analyze that predicting 5 future frames is optimum for our system. This trained LSTM is then integrated with the existing end-to-end system and its performance is evaluated against the system without the LSTM by validating results obtained on the DukeMTMC dataset. In this way we analyze its impact and present a qualitative study of why it does not improve the systemâs accuracy for some cameras in the dataset. We then fine-tune our trained LSTM model for each camera individually to observe the increase in the accuracy for every camera. This provides a proof of concept that we require an algorithm that remains specific to each camera and learn movement patterns for each specific perspective. We conclude the study by comparing our complete end-to-end system's performance with the state of the art.

DEDICATION

This work is dedicated to my family and my friends.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Hamed Tabkhi, who gave me a chance to work with him in the TeCSAR lab and always pushed me to give my 100 percent. I would also thank all my lab-mates who always helped me whenever I needed their help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AOS Average Overlap Score

FPS Frames Per Second

IDF1 Identification F1

IDP Identification Precision

IDR Identification Recall

IoU Intersection over Union

LSTM Long Short Term Memory

MTMCT Multi Target Multi Camera Tracking

Re-id Re-Identification

RNN Recurrent Neural Network

CHAPTER 1: INTRODUCTION

With the ever-increasing number of surveillance networks in urban areas, there has been a revolution in the computer vision community to develop algorithms for such systems which are not only accurate but real-time [2]. The primary reason for surveillance is to provide safety for civilians against criminal activities. In such cases it is important to eliminate any cloud computation which brings latency into the system and push maximum computation to the edge devices for immediate response [3, 4, 5]. Hence, there is a growing need for end-to-end Multi-Target Multi-Camera Tracking (MTMCT) algorithms which are not only accurate but also real-time. The pipeline of these MTMCT algorithms consists of a detection framework which detects people in the current scene, followed by a re-identification (re-id) algorithm which make sure that the detections have been correctly re-identified in the consecutive frames of the video stream, after which comes a tracking engine which is responsible to track these detected pedestrians by understanding their respective trajectories. Fig. 1.1 illustrates how a camera network detects and re-id pedestrians across cameras.

The tracking mechanism becomes an important part of the system as it helps with understanding the movement patterns of different people not only within the camera scope but also across multiple cameras. Moreover, as we understand movement behavior of these pedestrians we can predict their future trajectories and leverage them to deal with scenarios like occlusions and miss-detection which may occur during the detection phase.

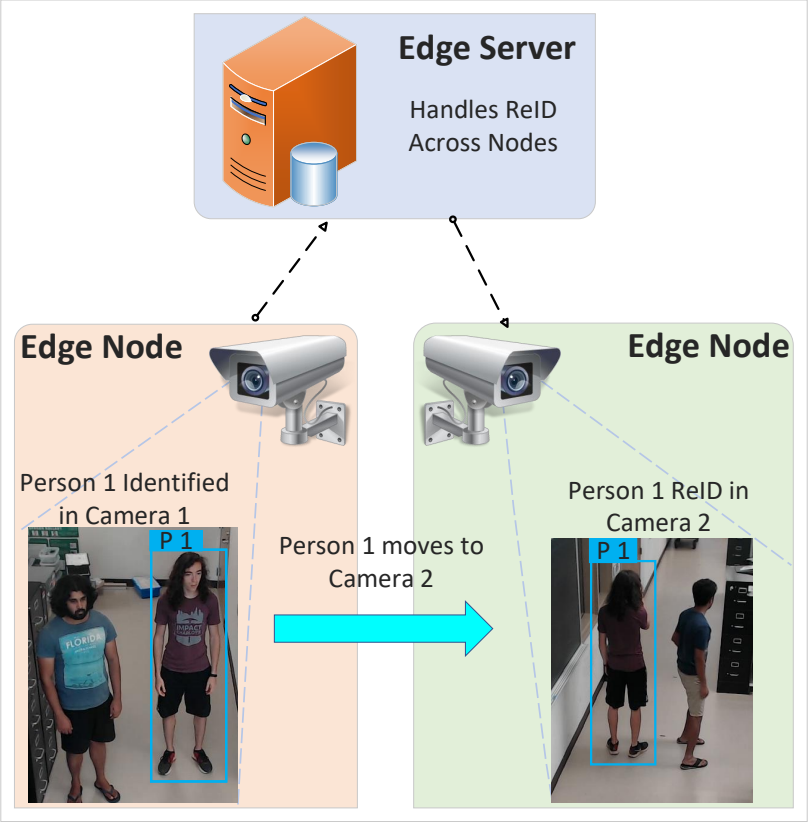

Figure 1.1: Detection and re-id across 2 cameras

1.1 Problem Statement

Executing this above mentioned end-to-end pipeline on the edge device with hardware and power constraints is a challenge but it is important as we need real-time awareness. Also, sending personal information to the cloud can compromise with the privacy of the people which is highly undesirable. Many algorithms track the objects by executing their re-id networks on the detections by running these algorithms on every frame of the video feed which can be accurate but introduce latency in the system. Most of the recently proposed trackers which have been evaluated on the MOT benchmark [6] achieve a minimal Frames Per Second (FPS) as their throughput. The primary reason for such a low throughput is large Convolution Neural Network (CNN) based detection and re-id networks which are highly compute intensive. As a result, we explored less compute intensive MTMCT networks which can run efficiently on the

edge devices yielding a decent accuracy as well as throughput as a complete end-to-end system. We came up with a solution similar to the current state of the art MTMCT framework, deepcc [7]. We use OpenPose [8] as our pedestrian detection network, a triplet loss network for carrying out the re-id process on the obtained detections. To make the system real-time and feasible on the edge device, we run OpenPose at a much lower network resolution. The triplet loss network uses MobileNet-v2 [9] as its feature extractor instead of ResNet-50 [10]. This end-to-end system when run on the NVIDIA AGX XAVIER [11] device yields a throughput of a little greater than 5 FPS. Although the throughput is good, this system lacks detection and re-id accuracy. An efficient tracker which can understand the movement behaviors of different people in the scene and can predict their future trajectories would be able to compensate some inaccuracies introduced in this real-time system.

1.2 Proposed Solution and Contributions

In this research, we present a coarse-grain LSTM-based tracker for our system to realize real-time edge video analytic. We utilize the pattern learning capabilities of LSTM to predict future trajectories of objects in the video scene at a granularity of 5 FPS. We envision to increase our existing system's accuracy by using these predictions in the instances of miss-detection and occlusions. We achieve this by training our LSTM module on single object sequences shot at 60 FPS, exhibiting different movement patterns, offline and then evaluating its performance when integrated into the system. Instead of using any large scale feature maps as the LSTM input like most of the other approaches we only use the key-point coordinates of the pedestrians obtained from the detection phase. This removes any background noise and only provides the key-features to the LSTM so that it can be trained smoothly. We make the following contributions in this study:

1 . Coarse-grain LSTM tracking framework with prediction over the next 5 frames for high frame rate videos (e.g. 60 FPS) - Details provided in section 4.2.

2 . Integration of proposed LSTM framework into our end-to-end pedestrian tracking pipeline to enhance pedestrian re-identification accuracy - Details provided in section 4.3.

3 . Leveraging transfer learning to specialize and fine-tune the pre-trained LSTM per each camera with the aim to enhance the re-identification accuracy with respect to each camera - Details provided in section 4.4.

4 . Detail performance analysis on re-identification by considering the impacts of static and dynamic occlusion - Details provided in section 5.5

5 . Identifying the optimum point in trade-off between (prediction steps and accuracy) in the proposed coarse-grain LSTM tracker - Details provided in section 5.4.1

6 . Overall, achieving pedestrian re-id accuracy within less than 3% of current state of the art [7], with 6 X power reduction - Details provided in section 5.7

CHAPTER 2: BACKGROUND

This chapter will go in detail for every part of the MTMCT pipeline described in chapter 1, pedestrian detection, re-id and tracking.

2.1 Pedestrian Detection and Pose Estimation

The first step in the task of tracking pedestrians is detecting them in the video stream. The video stream is broken down into discrete frames and each frame becomes an input to the detection algorithm. The task of detection algorithm is to localize different objects in that particular frame and classify these localized sections. Most of the detection frameworks localize by drawing a Bounding Box (BB) around the object of interest. Fig. 2.1 shows 2 people being detected in a frame of a video stream.

All the detection algorithms use CNNs at their core for extracting the most prominent features in the input frame. They learn to identify basic shapes, edges and similar characteristics in the frames as they are trained and likewise become accurate in classifying and localizing objects present in it and ultimately in the video. Based on the similar concept there has been a growing interest in developing pose estimation algorithms which try to localize different key-points of a human body for the pedestrians present in the frame. Fig. 2.2 shows the output of a pose estimation network on an input image.

In section 3.1, we will review some of the recent work in object detection and human pose estimation, and justify our use of OpenPose framework as our detection network.

2.2 Pedestrian Re-Identification and Tracking

The core of every MTMCT framework is an accurate re-id network which make sure that every detected pedestrian retain its acquired ID all throughout the video

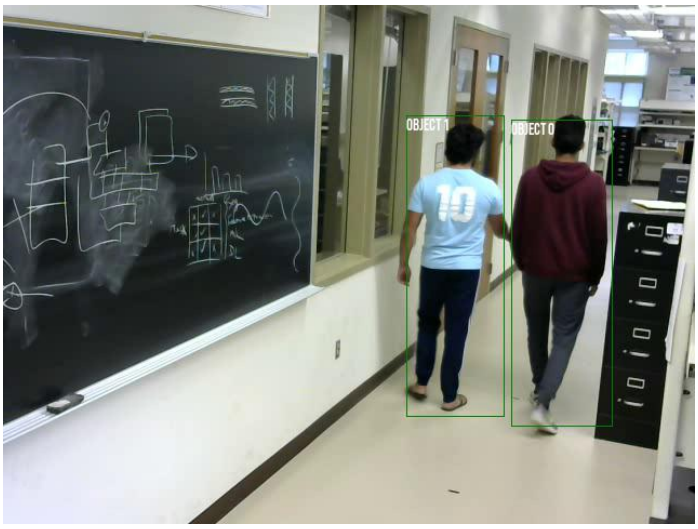

Figure 2.1: Object detection in a single frame

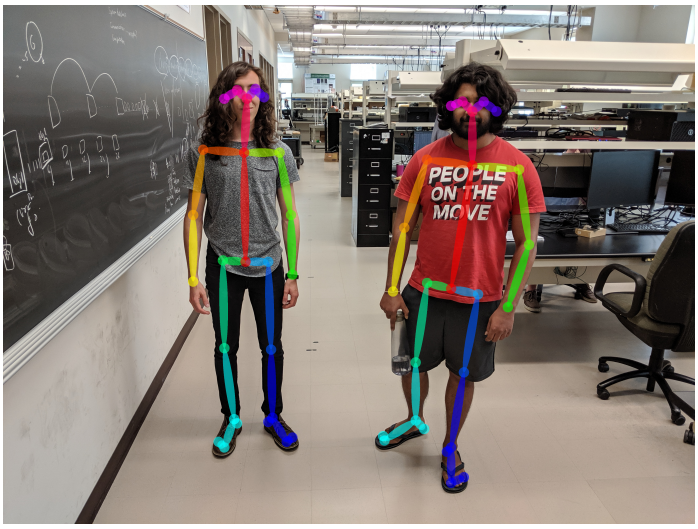

Figure 2.2: Pose estimation

stream. In this way we track them not only in the present camera view but across multiple cameras as well. Correctly re-identifying objects has been a long pursued research in the computer vision community. In Fig. 2.1 we can see that the 2 objects have been assigned IDs object0 and object1. Section 3.2 will go into the detail of the previous work done for efficient re-id and explain the reason of using triplet-loss network in our system which is described in section 2.3.

Tracking objects or specifically pedestrians implies that we correctly detect and

re-id them throughout the video stream. To accomplish this we need to run these algorithms on every frame of the video which is difficult in real-time as these networks are compute intensive. To make the system near real-time we use smaller detection and re-id networks, developed for embedded platforms, compromising on accuracy. To deal with this deficit in accuracy we explored algorithms which can understand movement patterns of different pedestrians and predict their future positions. By knowing their future positions we can deal with scenarios like miss-detections incurred by the detection algorithm and occlusions which is very frequent in real world cases.

LSTM [12], which is an improved form of Recurrent Neural Network (RNN), have been proven to achieve state of the art performance for pattern recognition problems [13, 14, 15]. They inherently have a memory cell which keeps track of all the relevant data they have seen and learn to recognize the hidden patterns in it. We leveraged these pattern learning capabilities of LSTM in this research and trained it to predict future positions of the pedestrians in the scene at a coarser granularity. A detailed explanation will be presented in section 4.2.

2.3 Triplet loss network

Fig. 2.3 shows the basic structure of a triplet-loss network as it trains 3 instances of a feature extractor to produce the most discriminating feature representations of a person. To effectively train the network we give it 3 different region crops (bounding box region) of people. One acts as an anchor or the ground truth crop, one is the positive sample or the same person as the anchor and one is a negative sample. We use the following loss function after the respective feature maps are extracted:

$$Loss = \sum_{i=1}^{n} \left[\alpha + f_i^a - f_i^{p2} - f_i^a - f_i^{n2}\right]_+, \tag{2.1}$$

where α is margin, f^a, f^p, and f^n are embedded appearance features of the anchor, positive, and negative samples for the class i, respectively. As the model is trained

and the loss is minimized all the samples from class i are forced to be inside the radius α in the higher dimensional space. The size of this new dimensional space is 1280, which is the size of the feature map extracted using the feature extractor.

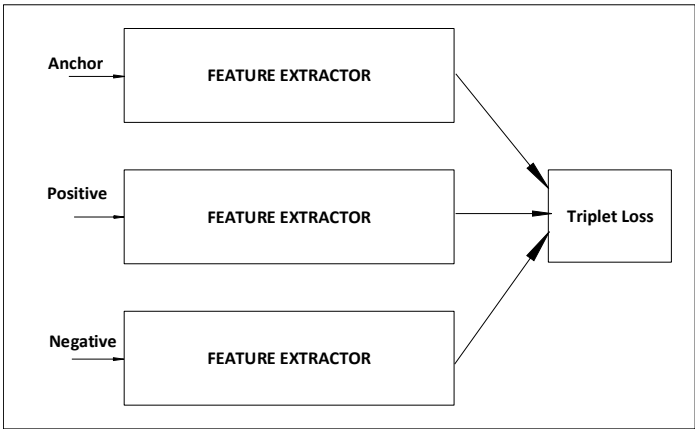

Figure 2.3: Triplet-loss network

After training we use a single instance of this feature extractor, which in our case is MobileNet-V2 [9]. The reason for using this network is that it was developed for embedded platforms. In this architecture, a regular convolution operation is broken down into depth-wise and point-wise convolution which drastically reduces the number of parameters for the network. Furthermore, adding linear bottleneck layers increases its performance. At run-time, we re-id people by calculating the euclidean distance between the features extracted in the current frame and the previous frame features. Feature maps extracted for the same person will have the minimum euclidean distance between them.

2.4 Key-Track

The coarse-grain LSTM presented in this work is based upon our previous work, key-track [1]. Fig. 2.4 shows the end-to-end architecture of key-track. Key-track is also a LSTM based tracker which was able to achieve good accuracy in predicting the next frame positions of pedestrians on the DukeMTMC dataset. It is a fine-grain tracker and worked for sequences at 60 FPS taking 18 key-points as its input

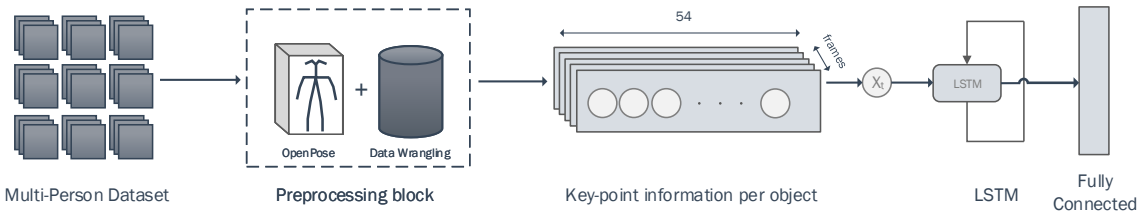

Figure 2.4: Key-track architecture [1].

provided by OpenPose. Since our existing system works at 5 FPS, we could not use key-track, instead we developed a coarse-grain key-track which works at 5FPS and predicts future positions way ahead in time. Another difference is that we used 25 body key-points for the coarse grain LSTM instead of 18 as in key-track. The reason being 25 key-points will provide the LSTM with more information to learn from.

2.5 Dataset Curation

The coarse-grain LSTM is trained on single pedestrian sequences but can infer on multiple pedestrians at run-time. For generating these single-pedestrian sequences and training our LSTM, we used the DukeMTMC dataset [16]. This dataset has 8 non-overlapping different camera views with a total of around 2834 people annotated. Total video footage is more than an hour and it is shot at 60 FPS with a resolution of 1080p inside the Duke University campus. We chose this dataset as it emulated the real world scenario of a surveillance setting.

For curating single pedestrian sequences we used the same concept used in [1] for data curation. We ran OpenPose with its body-25 model on each video of the dataset, obtaining key-points for every person in each frame of the dataset. We then mapped these key-points to the ground truths by randomly selecting an ID from the ground-truth file, iterating over every frame for that person, and checking if the key-point values lie in the ground-truth bounding box region. Using this technique we created a dataset containing only single pedestrian by isolating it from a multi-pedestrian dataset. Our coarse-grain LSTM was trained using these curated sequences(explained

in section 4.2).

CHAPTER 3: Related Work

In this chapter, we overview related research on pedestrian detection, re-identification, and tracking.

3.1 Pedestrian Detection

There have been many recent advancements for developing state of the art object detection algorithms. Single Short Detection (SSD) [17] is one such highly accurate network which only processes the image once in its pipeline. Similarly You Only Look Once (YOLOv3) [18] localizes around 2000 objects found in the COCO dataset [19] while staying computationally efficient. Faster R-CNN [20] works on the concept of region proposals, dividing a frame into different regions in order to classify them and finally localize. All these frameworks detect general objects whereas our specific focus is to track pedestrians so we reviewed some of the recent pose-estimation networks. DeepPose [21] by google uses cascaded Deep Neural Network (DNN) to regress poses to human body joints. The DeepCut [22] framework treats pose-estimation as an integer linear problem by simultaneously partitioning and labelling different body parts. A more recent network, OpenPose [8] uses part affinity fields between different body joints to estimate its pose. We decided to use OpenPose as it is accurate, invariant to the number of people in the scene and is able to achieve higher throughput (FPS) when compared to other recent works. Another advantage is that it can be customized to run at different network resolutions which can be utilized to explore different accuracy and throughput needs.

3.2 Pedestrian Re-identification

Classical computer vision approaches like those in [21] which are based on covariance descriptors augments various feature representations of an image like RGB, HSV, Local Binary Patterns, etc over a Mean Riemennian Martix(introduced by Bak et al. [23]) from multi-shot images to find similarities between different images have been done. Similar approaches were adopted in [24] for real-time embedded computation but the authors do not provide with any accuracy evaluation. In [25], Icaro et al. generates unique signature for each object which comprises interest points and color features for the object and calculate similarity between different signatures using Sum of Quadratic Differences(SQD). Similar classical approaches are demonstrated by [26] and [27], which uses Biologically Inspired features(BIF) and k-shortest path algorithm. Classical techniques are promising however with the boom in deep learning algorithms and plethora of computational power all thanks to top of the line GPUs, they are even surpassing human level recognition for re-id.

Modern deep learning techniques like Alignedreid[28], extract features from ROI using CNNs as base networks and then divide the feature map into local and global features intuitively dividing the ROI into horizontal sections and matching each section with the other images. Loss function used is triplet loss in this work and the authors claim to surpass human level performance for person re-id. Xiaoke et al. in [29], use videos instead of separate frames to learn the inter-video and intra-video distances between people in them effectively creating triplet pairs. Tong et al. [30] proposed an Online Instance Matching(OIM) loss paradigm which uses a Look Up Table(LUT) for labelled objects and a cicular queue for unlabelled objects and learns to re-id people on the go. Yantao et al. in [31] formulates the problem of person re-id into a graph neural network problem, with each node denoting a pair of images whose similarity and dissimilarities are learned through a message passing technique between the nodes. Siamese network is used to compute similarity metric between

pairs. Authors in [32] introduce spatiotemporal attention models to learn key spatial features of objects throughout the video. On similar lines the work in [33] also learns the spatial and temporal behavior of objects by translating the feature map of the ROI into adaptive body-action unit with each unit highlighting a prominent body part of a person. Ju et al.[34] use bidirectional Long short Term Memory(LSTM) neural network to learn the spatial and temporal behavior of people throughout video. They use two copies of the same network to learn generic and specific features of people and shortcut connections for smooth learning. Minxian et al. in [35] employ an unsupervised approach for person re-id. They generate labelled data with sparse space-time tracklet sampling by observing the spatial and temporal arrangement of people in a video scene and train the network to discriminate between single-camera tracklets whereas associate cross-camera tracklets.

Almost all the aforementioned work is novel and many of them achieve state of the art performance, however they all use very complicated and deep networks which would hinder their performance in real-time scenarios. Instead we chose a light-weight feature extractor for our system for doing re-id, which was trained using a triplet-loss network(explained in section 2.3).

3.3 Pedestrian Tracking

Works like [36] used kalman filters to understand movement behavior of objects. Kalman filters might prove to be useful but they do not account for any previous movement patterns seen in the objects. Authors in [37] used RNNs for object tracking and they show good results. Work like [38] also proved how RNNs can understand patterns in the movement of objects for tracking them. Recent research like [39] and [40] used LSTM coupled with a detection framework for tracking objects. Authors in [40] used YOLOv1 as their detection algorithm and LSTM to predict future positions for single object sequences. They re-use the feature vector extracted by their detection framework as the input to their LSTM. Although they get good results, the size of

the feature vector os 4096, which makes their LSTM model too big. Milan et al. in [6] used an online tracking method with LSTM to track multiple objects in the scene. The authors record real-time performance but at the expense of accuracy.

All the mentioned work that has been done using LSTM for tracking uses some kind of feature vector as its input. While it may help the LSTM to understand the visual context of the object, it also contains background noise as the feature has been extracted for the region inside the bounding box. In this work, instead of using a feature vector as the input for the LSTM, we used the body key-points that is given by the OpenPose framework as part of its detection phase. In this way, we make sure to provide only prominent information about the object to the LSTM. As a result the input size reduces to 75, giving us a light-weight model which can easily be ported to any embedded platform.

CHAPTER 4: SYSTEM DESIGN

In this chapter we will go into the details of how our current system works. First in section 4.1 we will explain our existing end-to-end system with the detection and the re-id algorithm and how it tracks pedestrians for each camera as the edge node. Then in section 4.2 we will go into the detail of our coarse-grain LSTM tracker and how it was trained in order to predict future positions of the pedestrians in the scene. Section 4.3 will discuss the methodology of integrating this trained LSTM model into the existing system and how we utilized it to deal with miss-detection and occlusions. In section 4.4 we will explain how we used our trained coarse-grain LSTM for transfer learning by fine-tuning it on each camera separately.

4.1 System Overview and Limitations

Fig. 4.1 shows the overview of our system without the LSTM tracker. In this system we rely on the detection and the re-id algorithms to correctly track different pedestrians in the video stream. We fetch a frame from the video and send it to the OpenPose network to get key-points for the people in it. We then draw the bounding box for each person taking the minimum and the maximum key-point value, after which comes the re-id phase. We then carry out a spatial filtering by checking the Intersection over Union(IoU) for the current bounding boxes with the previously saved bounding boxes of the objects already re-identified. This provides a sanity check as an object is supposed to be near its previous position in consecutive frames. If the IoU is greater than zero, we send these candidates for feature extraction.

We crop the bounding box regions for these spatially filtered pedestrians and batch them to extract their feature maps. We use MobileNet-V2 as explained in section 2.3

to obtain feature maps of size 1280. We now calculate the euclidean distance between these new feature maps and the previous object's feature maps to get a similarity metric. Similar people will have the lowest euclidean distance between their feature maps. In this way we get the best matches for the previous people in the scene. The matched pedestrians are then given the same ID as their previous counterparts and saved in the local database.

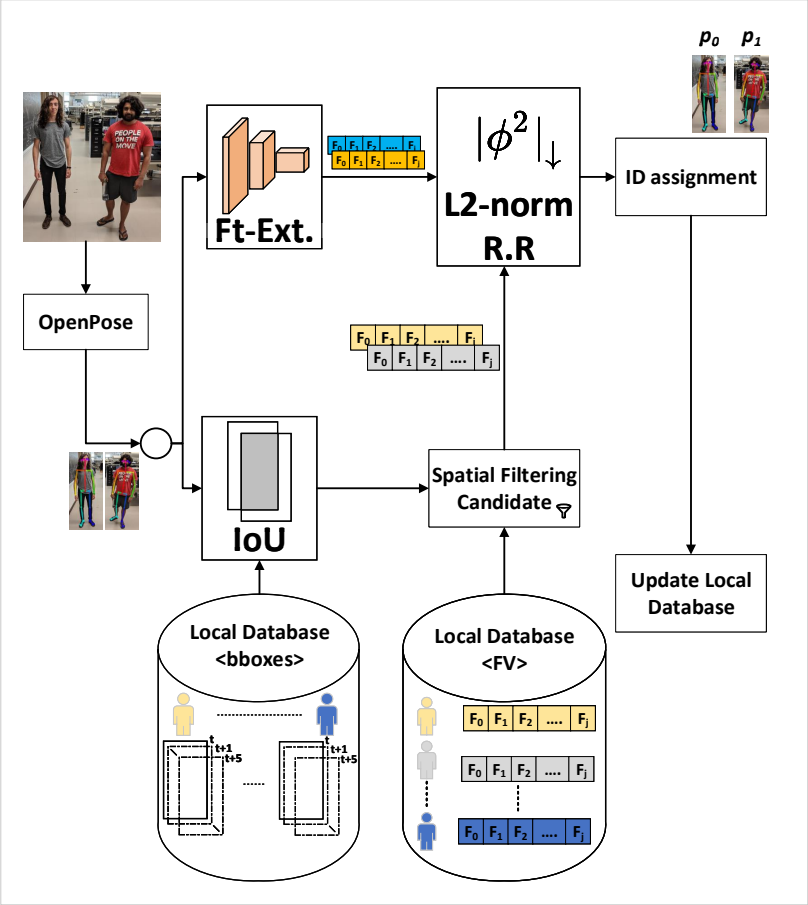

Figure 4.1: Overview of the system

The local database is where we save all the bounding box coordinates, feature maps and IDs for all the pedestrians that have been detected as well as re-identified. Although this approach works well, it suffers from some limitations in real-world scenarios.

Real world cases have instances of frequent occlusion. These can be static, dynamic,

short-term or long-term. Also, as OpenPose is running at a lower network resolution, it sometimes suffer from miss-detection for some pedestrians. This impacts the re-id accuracy of the system as in such cases the object disappears from its previous vicinity, as a result the spatial filtering fails. To overcome these limitations, we came up with the idea of using a coarse-grain LSTM to predict future positions of every pedestrian, so that these instances of miss-detection and occlusion can be dealt with. In the next section we go into the detail of our LSTM tracker and how it is trained.

4.2 The Proposed Coarse-Grain LSTM tracker

Training the LSTM to track multiple objects is a challenge in itself. Each object needs to be re-identified before we feed any input to the LSTM. The reason being we need some previous history for every object as the input. This previous history can also be called the time-step for the LSTM, as it learns from the previous patterns of every object. We could re-id these objects or pedestrians by mapping them to the ground truths but that assumes we have perfect re-id paradigm which is not the case in our real system.

We then came up with the solution of training it on single pedestrian sequences but doing inference on multiple pedestrians by batching them together in the system. In this way we re-use the same model parameters for every pedestrian. The method we used to curate these single pedestrian sequences is discussed in section 2.5.

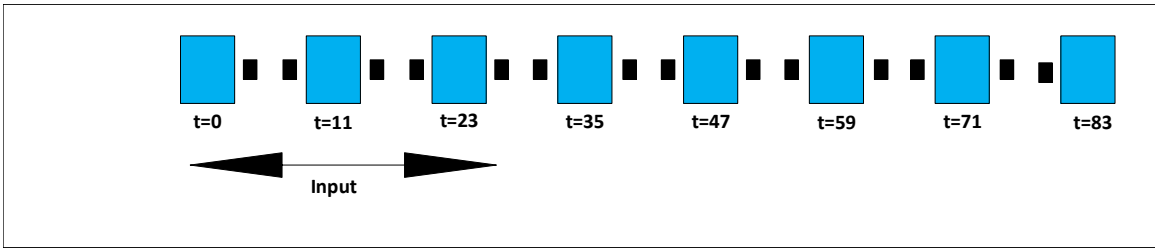

Figure 4.2: Granularity of data for our coarse-grain LSTM tracker

These single pedestrian sequences that we curated are shot at a frame rate of 60 FPS but our existing system works at 5 FPS. This is the reason we had to choose a

coarser frame rate of 5 FPS for the input as well as predictions of this LSTM module. Fig. 4.2 shows an intuitive visualization of this transformation by taking an example of a single sequence. We start from t=0, and go till t=83 for the first training window in the sequence. We use a time-step of 3 for the coarse grain LSTM as it proved to be the best in [1]. So the first 3 frames in the coarse grain sequence at t=0,t=11 and t=23 become the input to the coarse-grain LSTM. The next 5 frames in this window are then predicted by the LSTM. In section 5.4.1 we explain the reason of choosing only 5 future frames for this coarse-grain LSTM. In this way, this window slides by 1 unit as the training goes on. The next window will start from t=1 and go till t=84 and so on.

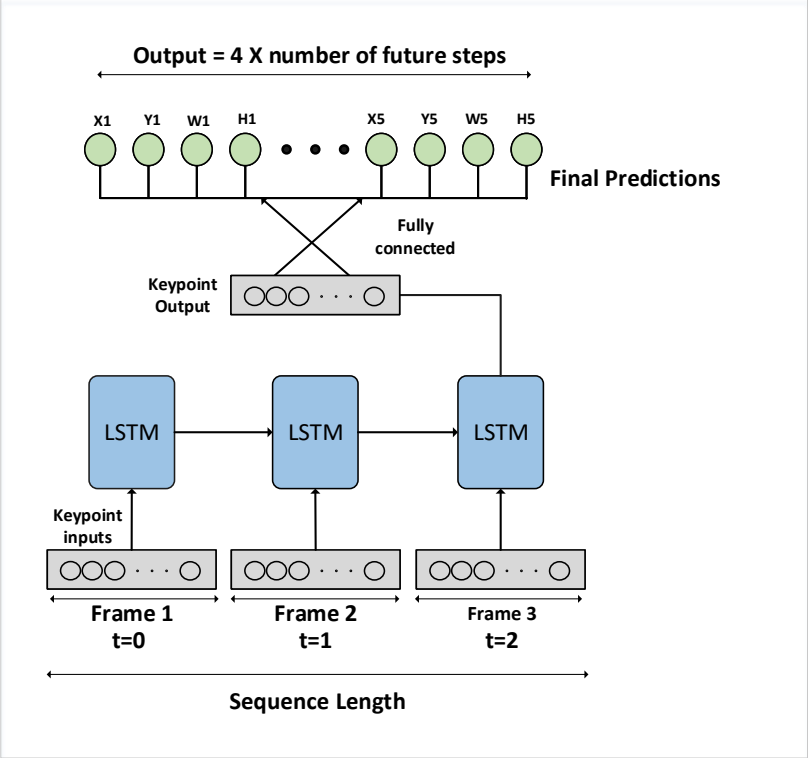

Figure 4.3: LSTM tracker

Fig. 4.3 shows the detailed structure of our LSTM tracking module. We use a time-step of 3 for the LSTM layer after which comes a fully connected layer with the output nodes equal to the future step times 4 (centroid coordinates, width, height) bounding box coordinates. The input of the LSTM is the key-point coordinates(25)

and the confidence value for each coordinate for each pedestrian. This makes the input size as 75 which is much smaller than any feature vector size given by any recent feature extractor.

Since our system runs at 5 FPS and our dataset from which we curated single pedestrian sequences was shot at 60 FPS, we pick every 12^{th} frame from the sequence. We append the key-point inputs from 3 consecutive frames for that person and make it the input to the LSTM. The last LSTM cell output is grabbed and fed into the fully connected layer which encodes this input into the next five bounding boxes to be in the sequence at the same granularity as the input. Mean-squared error is used as the loss function between these predictions and the ground-truth values with adam optimizer to minimize this loss using back-propagation. The whole sequence is traversed in this sliding window fashion till the last step prediction i.e. the 5^{th} step reaches the end of that sequence. We then move on to the next one and go through every sequence one by one. This constitutes 1 of the 150 epoches for which this coarse-grain LSTM is trained. The learning rate for training was initialized at 10^{-6}.

In this way our coarse-grain LSTM was trained to track pedestrians using just single pedestrian sequences from DukeMTMC. In the following section we will explain how this tracker was integrated into the existing end-to-end system.

4.3 End-to-End System integration

Fig. 4.4 shows our system with the LSTM module integrated with it. As explained in section 4.1 after getting the matches using spatial filtering and calculating euclidean distance between feature maps, we utilize LSTM predictions. As we get the matches, we check for any miss-detection for any existing match from previous frame. If there is a miss-detection, we check if we have LSTM predictions for that pedestrian in the local table. If yes, we use its next LSTM predictions as its detection and update the position of that missing pedestrian in the local table with its prediction. In this

way we check if that pedestrian re-appears using spatial filtering in every frame. If it re-appears, it would be in the vicinity of its prediction, which makes it a candidate for feature extraction and euclidean distance calculation. It is likely to have the smallest euclidean distance with its previously saved feature map before it disappeared. Fig. 4.5 explains this concept in more vivid detail in case of an occluded pedestrian. As a pedestrian gets occluded, we miss its detection from OpenPose. We then grab its LSTM predictions by iterating over them for each miss-detected frame, regarding it as an estimated position for the occluded pedestrian. As the occluded pedestrian re-appears it matches with its estimated position via spatial filtering and euclidean distance calculation retaining its previous ID after occlusion.

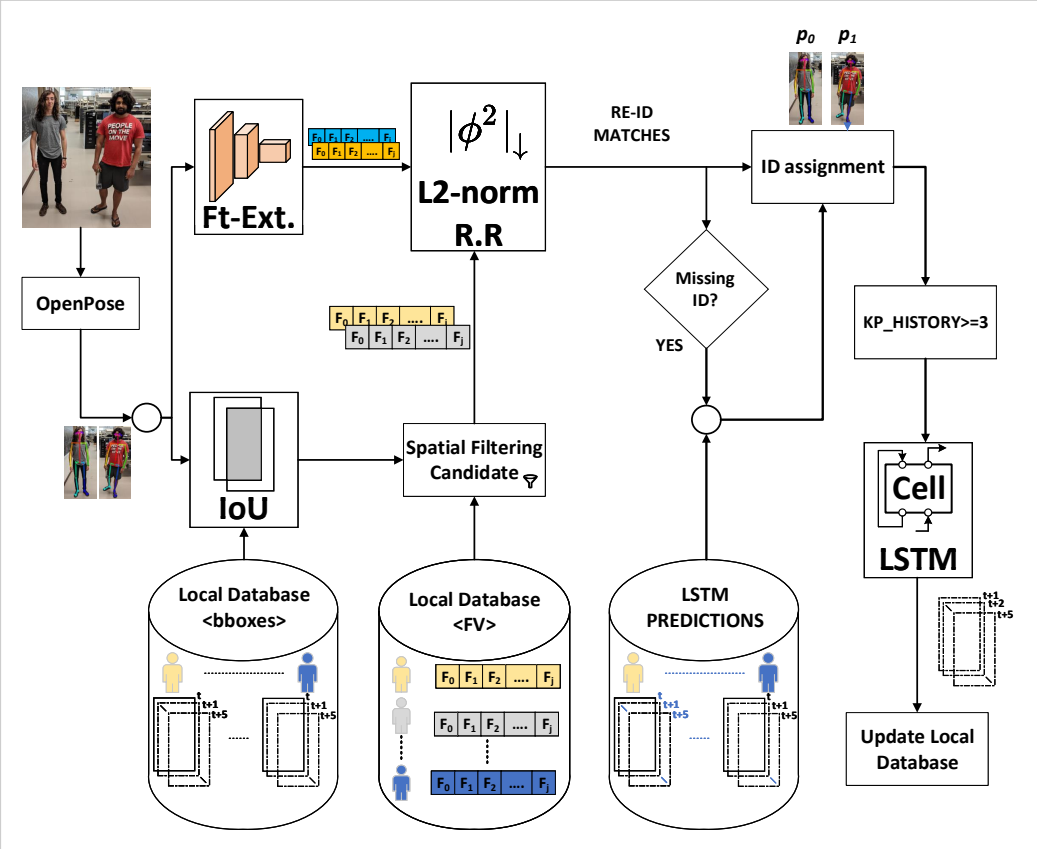

Figure 4.4: System with the LSTM tracker

After we assign IDs to all matched pedestrians and LSTM predictions for missing matches, we check for the `kp_history` which is the key-point history for each match.

Since we use a time-step of 3 for our LSTM tracker, if a match contains 3 previous key-points it becomes a candidate to get its future predictions from the LSTM. So, we batch the previous 3 key-points for all these candidate matches and send it to the LSTM as the input, getting their future predictions. These predictions are then saved into the local database, so that they can be used in cases of miss-detection and occlusion for these matches in the coming frames.

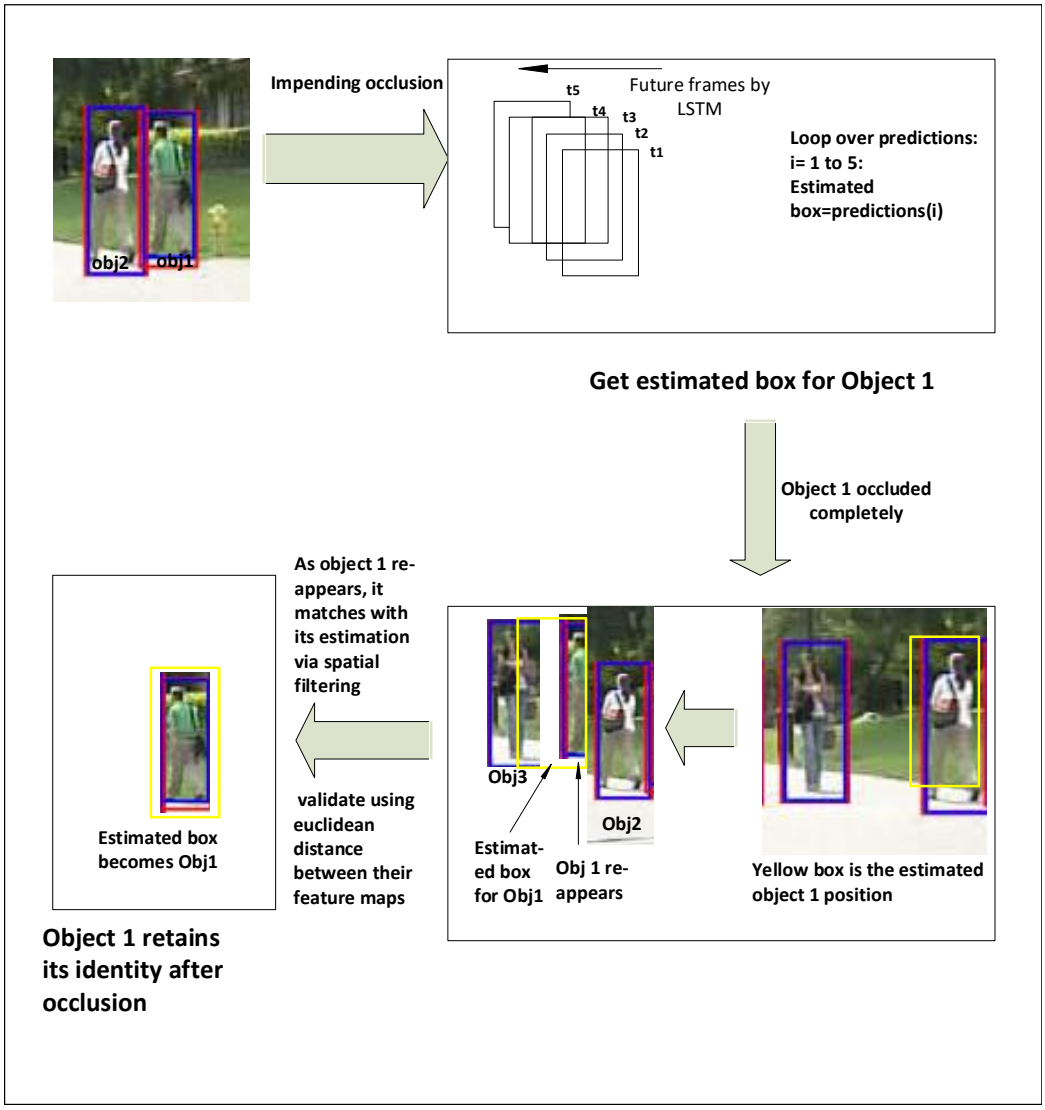

Figure 4.5: Using LSTM prediction with occlusion

Using this paradigm, we deal with scenarios like occlusion and miss-detections from OpenPose leveraging future LSTM predictions. We end up having more true positives

for ID assignments, which increases the system's accuracy(discussed in section 5.5). In the next section, we will see how we can enhance this trained coarse-grain LSTM using transfer learning to further increase the system's accuracy.

4.4 Enhanced Coarse-Grain LSTM with Transfer Learning

After observing the impact of the coarse-grain LSTM in the system, we see gain in the accuracy in all but 2 cameras. There was a slight decline in the accuracy of 2 cameras. After a qualitative analysis (section 5.5.1) for these cameras we hypothesize that we require specialized algorithms for every camera to individually learn the movement of pedestrians for that specific perspective. To provide a proof of concept for this hypothesis we fine-tune our trained coarse-grain LSTM using transfer learning for each camera separately. Fig. 4.6 shows the schematic for enhancing our coarse-

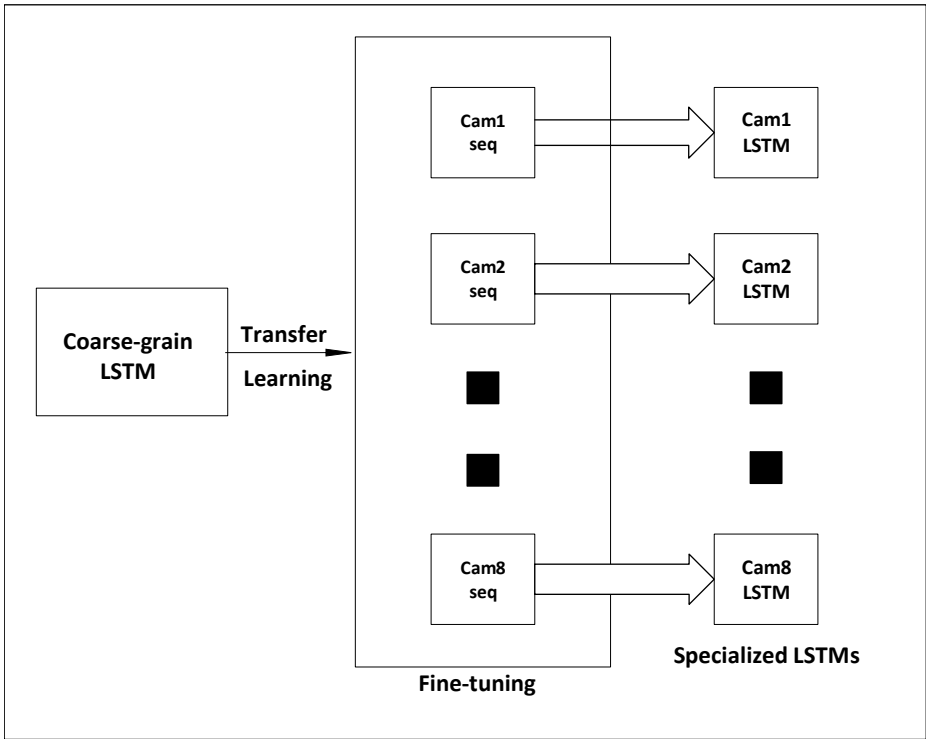

Figure 4.6: Using coarse-grain LSTM for transfer learning.

grain LSTM module using transfer learning for each camera. We take our coarse-grain LSTM model and fine-tune it for every camera using 45 sequences each. After this

process we get 8 different LSTM models specialized for each of these 8 cameras. We used the same mean-squared error like we used for the coarse-grain LSTM for calculating the loss while fine-tuning. Adam optimizer was used to minimize this loss with a learning rate of 10^{-7} for 80 epochs.

Each of the specialized LSTM models obtained after fine-tuning were used for their respective cameras in our end-to-end system. We show these results in section 5.6.

CHAPTER 5: SYSTEM EVALUATION

This section will go through the evaluation of our system, explaining the dataset we used, evaluation metrics, our setup for this evaluation followed by quantitative and qualitative analysis of the results.

5.1 Dataset

As explained in section 2.5 we use DukeMTMC for our system's evaluation. It is a MTMCT dataset which was shot at the Duke University campus using 8 cameras with non-overlapping views.

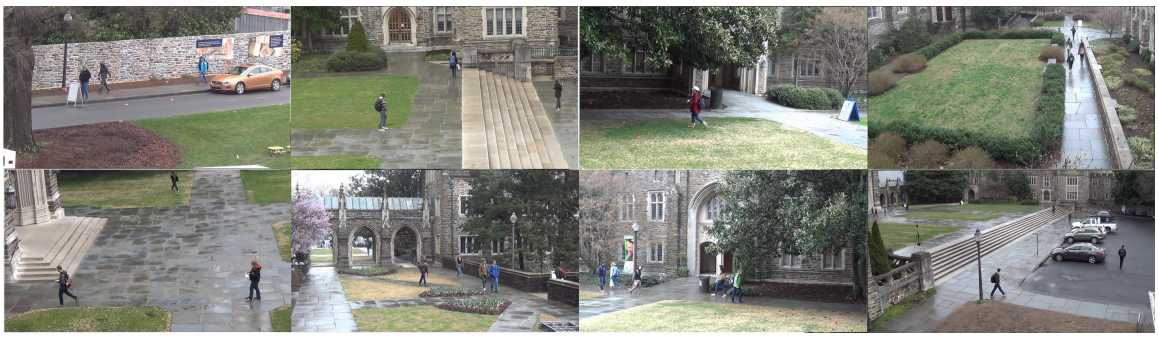

Figure 5.1: Different camera angles in DukeMTMC

Fig. 5.1 shows all 8 camera views that the DukeMTMC dataset offers. It has around 1800 instances of occlusion making it a challenging dataset. This dataset poses many real world problems making it a good validation test for the system built for pedestrian surveillance.

5.2 Evaluation metrics

To evaluate the performance of our LSTM module, we use region overlap or the Average Overlap Score (AOS). AOS is calculated using the Intersection over Union (IoU) method. It is defined as the ratio of the intersection area between the ground

truth bounding box and the predicted bounding box of the pedestrian to the union between the ground truth and the predicted box. The IoU for every frame is calculated for which the prediction is made and then averaged to get the AOS score.

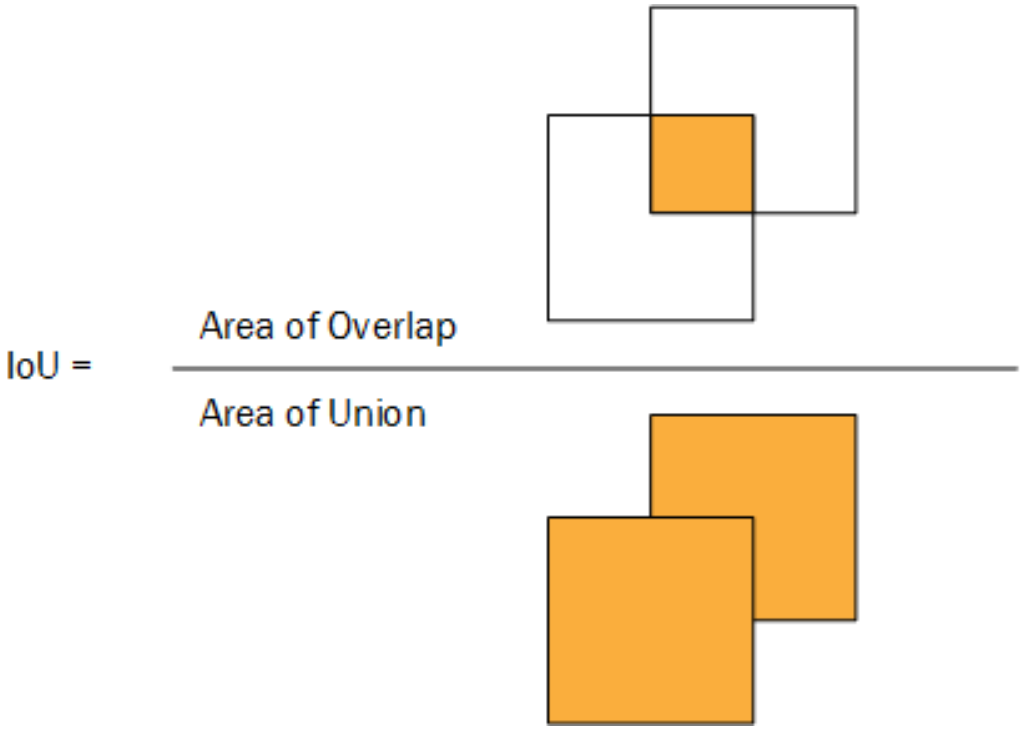

Figure 5.2: Intersection over Union (IoU)

For evaluating the system, we use the same evaluation metrics proposed in [16] for measuring re-id accuracy. Identification Precision (IDP), Identification Recall (IDR) and Identification F1 score (IDF1) are the 3 metrics we used to validate our results for re-id. To explain these metric, we first define true positives, false positives and false negatives and what they mean. We say a re-id is a true positive if it was re-identified correctly and the same ID is present in the ground truth. It is a false positive if we re-id the pedestrian but its not present in the ground truth. A false negative is when we miss a target. We then define True Positives (TP), False Positives (FP) and False Negatives (FN) as the sum of true positive, false positive and false negative for all frames. The IDP and IDR are calculated as IDP=TP/(TP+FP) and IDR=TP/(TP+FN). IDF1 is simply the harmonic mean of IDP and IDR giving a

measure of accuracy.

5.3 Experimental setup

The LSTM module was implemented and trained using PyTorch 1.0.1. For training we used the NVIDIA Tesla v100 GPU. The LSTM was trained on total 120 single object sequences (15 from each camera) curated from the DukeMTMC dataset and tested on 24 sequences (3 from each camera). To validate the accuracy of the full system with the LSTM module, the complete system functionality was ported to MATLAB and made into a simulation testbed to evaluate results. For these experiments, we used videos of all 8 cameras from the DukeMTMC dataset and validated our results on the `trainval_mini` frame set from the dataset. We computed detection misses following the truth-to-matching method with 0.3 IoU threshold with the ground truth. This MATLAB testbed was run on a x86 machine.

5.4 Coarse-Grain LSTM Tracker

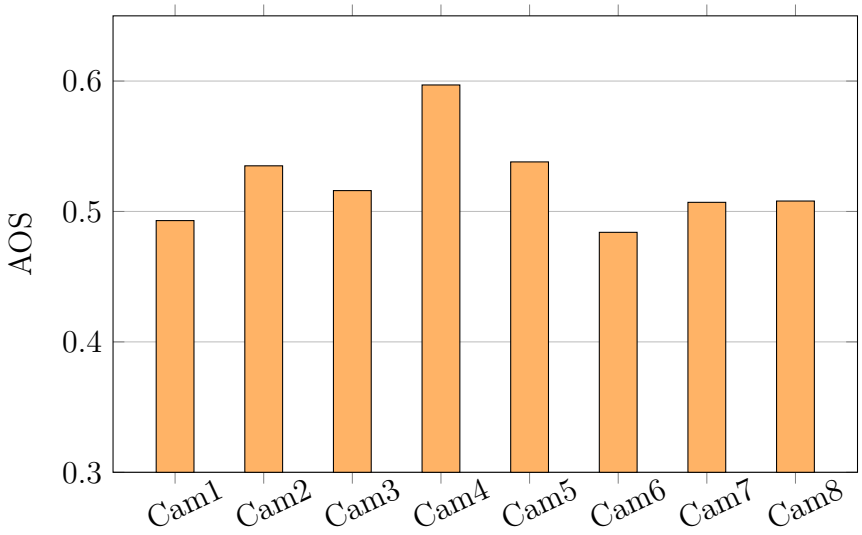

Figure 5.3: Average IoU for each camera on the testing sequences for first future frame.

Fig. 5.3 shows the AOS achieved by the LSTM between the first future prediction and the corresponding ground truth values for all 8 cameras on the test set. We

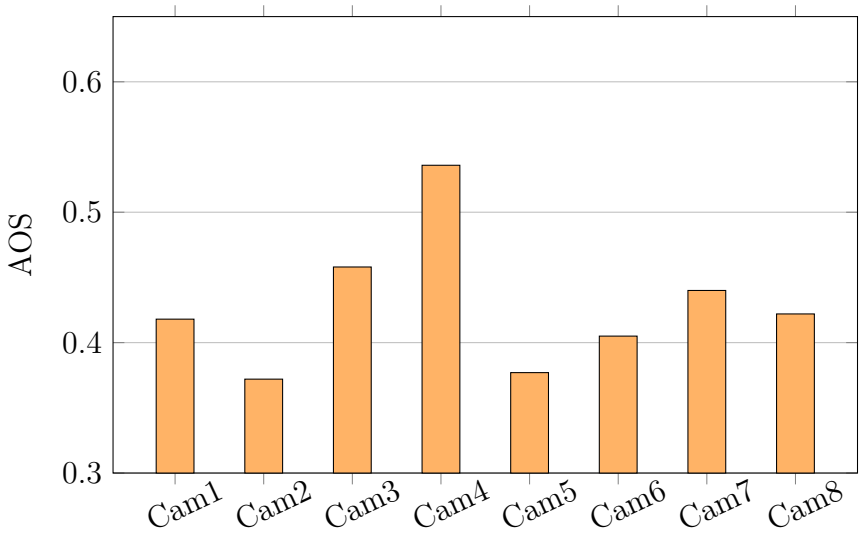

Figure 5.4: Average IoU for each camera on the testing sequences for third future frame.

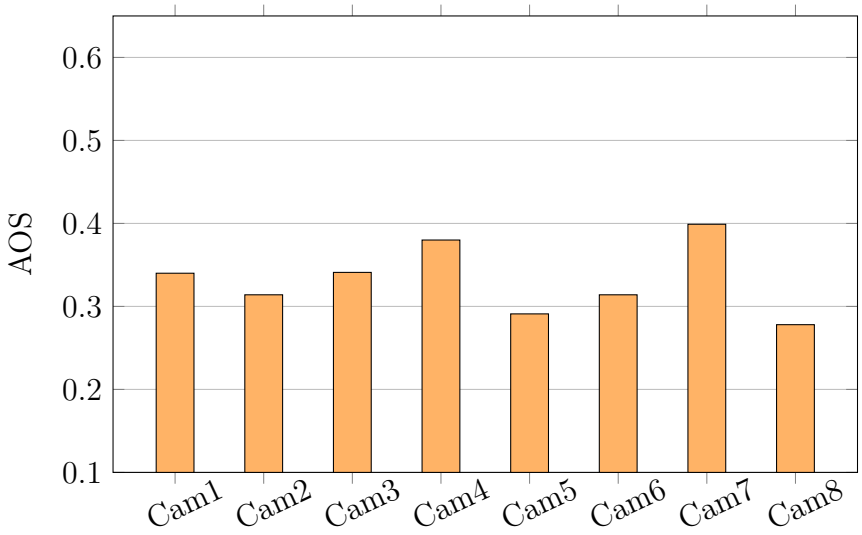

Figure 5.5: Average IoU for each camera on the testing sequences for fifth future frame.

observe the AOS around 0.5 for all cameras except for camera 4 which almost touches 0.6. Fig. 5.4 shows the AOS for the same cameras on the same test set but for the third future prediction. We see a drop in the AOS by almost 0.1. We then observe the AOS for the fifth future prediction on the same test set in fig. 5.5 to see further decrease in the AOS. In the next section we shall discuss this trend.

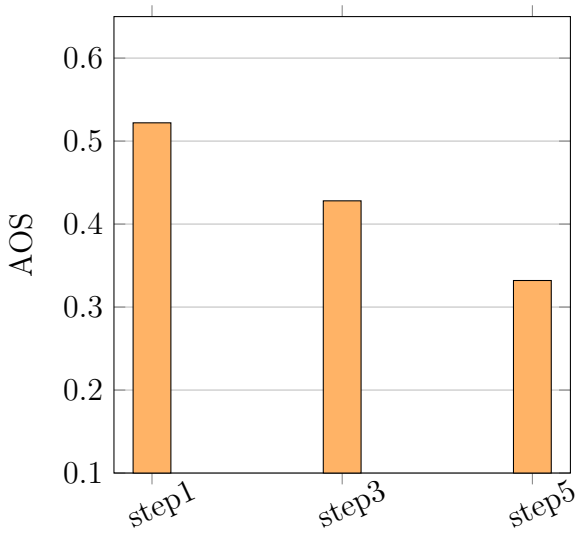

Figure 5.6: Average IoU comparison for future steps

5.4.1 Qualitative analysis

Fig. 5.6 shows the average AOS for all 8 cameras for the first, third and fifth future prediction step. We observe a linear decrease in the AOS as we go further into the future for predicting positions of different pedestrians. For step5 we get a little more than 0.3 AOS, which exactly matches our truth-to-matching IoU threshold of 0.3. If we go any further in predicting the future positions, we will go lower than this threshold which will affect the accuracy of the system. This is the reason we chose to predict only 5 frames ahead in the future.

5.5 End-to-End System

Fig. 5.7 shows the comparison of the F1 scores obtained with the system with no LSTM and with LSTM on the DukeMTMC dataset on all 8 cameras and the average change in F1. Although, it increases overall average by about 2 percent, it slightly decreases the F1 score for camera 2 and 7. In order to find out this discrepancy, we dive deeper into the change in the IDP and IDR for these cameras and try to correlate this analysis by visualizing these camera perspectives. In the next section, we will analyze this behavior and try to understand why the LSTM tracking does not impact

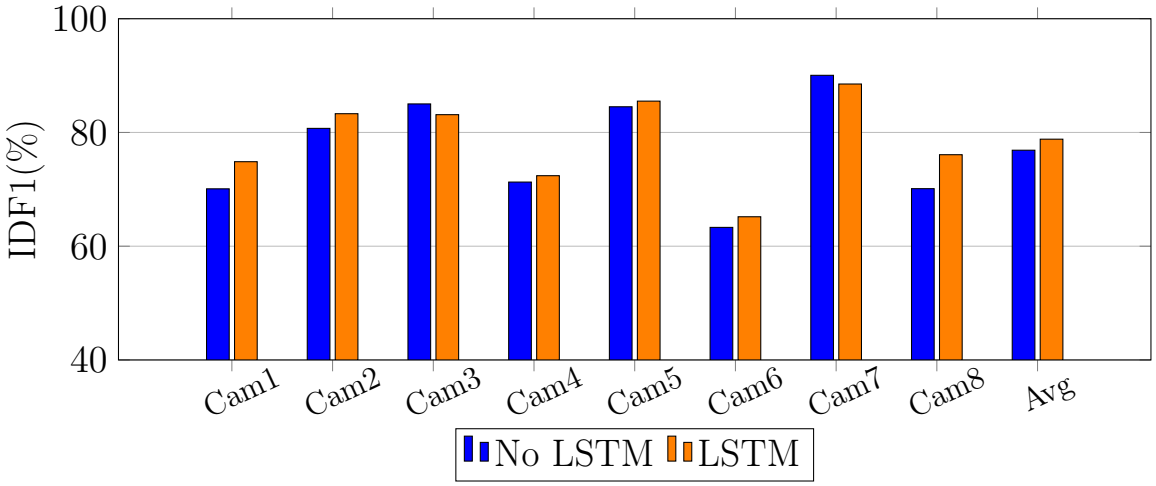

Figure 5.7: IDF1 Results for Single Camera

the performance in these cameras.

5.5.1 Qualitative analysis

Fig. 5.8 shows 3 consecutive frames from camera 3 from the DukeMTMC dataset. Green boxes are the ground truths, red boxes shows the detections and the blue boxes ate the LSTM predictions. If we observe the pedestrian highlighted by the green arrow, we see that in the first frame it has all ground truth, detection and LSTM prediction boxes. However as it approaches the door in the second frame, the ground truth box disappears but we still have the detection and the predictions. In the third frame as the object disappears, we still get the predictions from the previous frame as the system recognizes this scenario as a miss-detection. The system works with the prediction as a detection as it should be doing. This results in increased false positives which decreases the overall re-id accuracy of the system. Similarly in Fig. 5.9, which is a frame from camera 7, false positives would be incurred for any pedestrian that exits from the door in the middle of the scene.

In order to quantify this analysis, we compare the change in the IDP for the system with no LSTM and the one with LSTM. Fig. 5.10 shows this comparison. As we can see, there is a decrease in the overall re-id precision for camera 3 and camera 7. Since,

IDP is the ratio of true positive with respect to the sum of true positive and false positive, this analysis shows that false positives increase for these cameras with LSTM predictions.

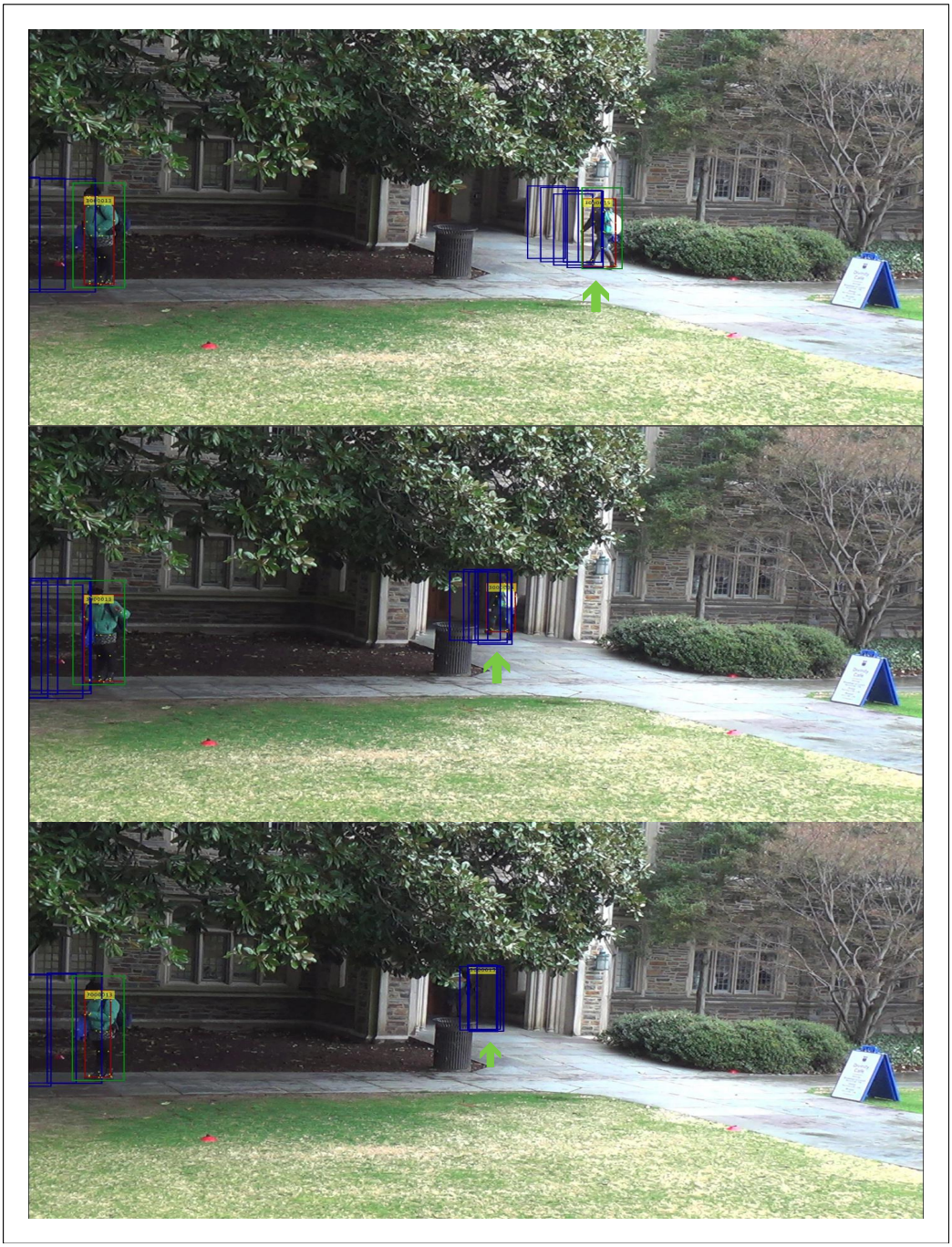

Figure 5.8: Frames from camera 3

Such kind of instances of static occlusions where a pedestrian exits the scene from the middle are tough to detect when the LSTM is trained on sequences from all the cameras. This gives us the intuition that the LSTM is not able to generalize based on every camera perspective. We require some kind of adaptive algorithm which learns from each camera's perspective separately and learns to not predict for such cases of static occlusion.

Figure 5.9: Frame from camera 7

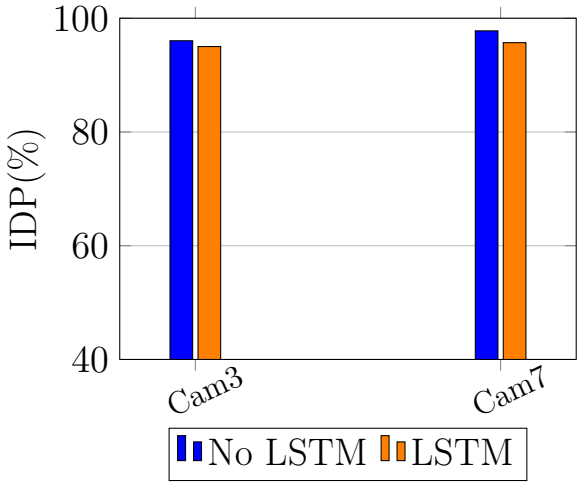

Figure 5.10: IDP for camera 3 and camera 7.

5.6 Enhanced Coarse-Grain LSTM with Transfer Learning

As explained in section 4.4, we fine tune our coarse-grain LSTM to specialize for each camera. After obtaining 8 different fine-tuned models for each camera, we evaluated our system's performance using these models.

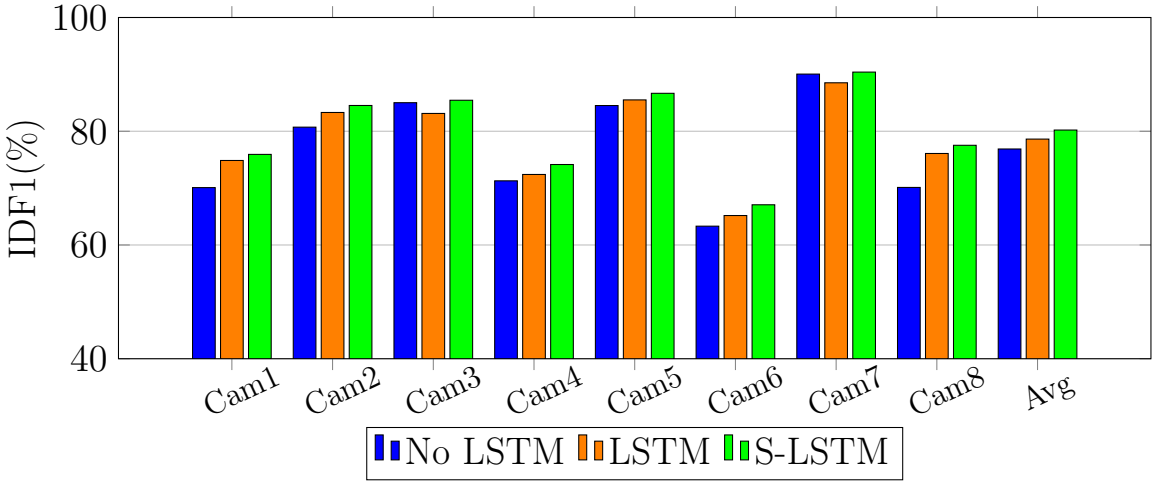

Figure 5.11: IDF1 Results for Single Camera

Fig. 5.11 shows the IDF1 results for each camera with no LSTM, with LSTM and with specialized LSTM (S-LSTM). As we can see there is an increase in the F1 score for all cameras when compared to a general LSTM. We observe the overall increase in the average F1 score by around 4 percent. This gives us a proof of concept that we do need specialized algorithm which understands the scene from each camera perspective and learns to predict movements of pedestrians for that specific scene setting.

5.7 Multi-Camera and Comparison with state of the art

In this section we will compare our system's performance with the state of the art which is DeepCC[7]. To compare the algorithm performance we use our specialized coarse-grain LSTM modules integrated in our system. We compare per camera F1 score as well as multi-camera performance for evaluating re-id accuracy across all 8 cameras.

Fig. 5.12 shows the comparison of re-id accuracy of our end-to-end system and

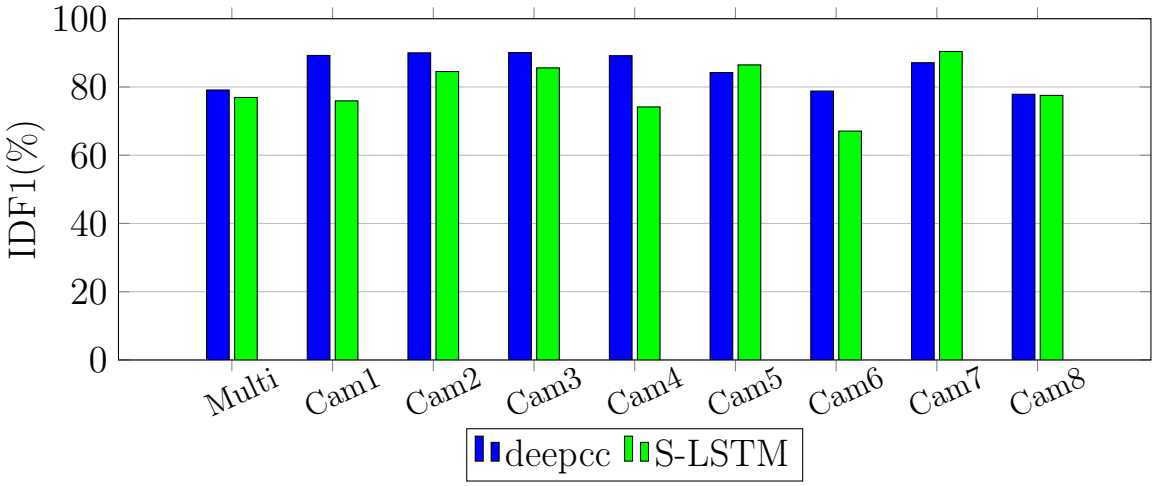

Figure 5.12: IDF1 Results for Multi-Camera Single Camera

DeepCC. As can be seen, we are about less than 3 percent short of the multi-camera F1 score when compared with the state of the art.

We now compare both of these systems in terms of throughput or FPS and the power consumed by these systems. For all measurements our end-to-end system is run in real-time on the NVIDIA AGX XAVIER(xavier) embedded platform. DeepCC is measured using OpenPose running at maximum accuracy configuration for detection and for re-id we use ResNet-50 as described in [7].

Table 5.1: FPS and Power Consumption of Real-Time Inference

System	Our system	DeepCC	DeepCC	DeepCC
Device	Xavier	Titan V	2xTitan V	V100
FPS	5.7	2.5	4.7	2.7
Power	25.01W	155W	320W	179W
Detailed Xavier Power Consumption				
Device	GPU	DDR	SOC	Total
Power	19.49W	2.734W	2.781W	25.01W

Table 5.1 shows the FPS and power results obtained for our system when compared to DeepCC. On the xavier platform we achieve 5.7 FPS while on a NVIDIA titan v DeepCC achieves 2.5 FPS as its throughput. Our system consumes just 25 watts

while DeepCC requires 155 watts which is 6 times the power our system uses. We also present the breakdown of the power consumed by our end-to-end system on the xavier.

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this research, we measured and analyzed the impact of tracking by using LSTM in our existing end-to-end system. This coarse-grain LSTM which was trained at 5 FPS and was optimally chosen to predict 5 future frames. On evaluation, it was able to increase the accuracy of all but 2 cameras. After providing an analysis, we envisioned separate algorithms for every camera which can learn instances of static occlusions and movement of pedestrians in isolation for each camera. To present a proof of concept we used transfer learning to fine-tune our coarse grain LSTM and observed an increase in accuracy for every camera. We then compared our system performance with these specialized LSTM modules with the state of the art, DeepCC and found that we achieved within less than 3 % accuracy for multi-camera while consuming less than 6 times the power.

This above analysis proves that our system is appropriate for real-time edge analytic.

Some of the future work we have in mind:

1 . Although specialized LSTMs work well in our existing system, their use is not optimum. We need to develop some adaptive algorithm which adapts to a particular camera perspective and learns its scene setting in an online learning paradigm.

2 . We envision to increase the throughput of our system and make it real-time. One way would be to restrict the execution of the detection network, which is a bottleneck, for some frames in between, and use the LSTM predictions for these intermediate frames.

3 . Extend the impact of these future trajectory predictions beyond a single camera to a multi-camera setting by analyzing variation in speed of different pedestrians and predict their appearance in the next camera as they leave their present camera view.

REFERENCES

[1] P. Kulkarni, S. Mohan, S. Rogers, and H. Tabkhi, "Key-track: A lightweight scalable lstm-based pedestrian tracker for surveillance systems," in *Lecture Notes in Computer Science*, Springer International Publishing, 2019.

[2] M. Sapienza, E. Guardo, M. Cavallo, G. L. Torre, G. Leombruno, and O. Tomarchio, "Solving critical events through mobile edge computing: An approach for smart cities," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–5, May 2016.

[3] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2016.

[4] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, pp. 84–91, Oct 2016.

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, Oct 2009.

[6] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[7] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6036–6046, 2018.

[8] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1302–1310, July 2017.

[9] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 2704–2713, 2018.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

[11] M. Ditty, A. Karandikar, and D. Reed, "Nvidia xavier soc," Aug 2018.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] P. Lin, X. Mo, G. Lin, L. Ling, T. Wei, and W. Luo, "A news-driven recurrent neural network for market volatility prediction," in *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 776–781, Nov 2017.

[14] A. Ray, S. Rajeswar, and S. Chaudhury, "Text recognition using deep blstm networks," in *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, pp. 1–6, Jan 2015.

[15] D. H. Oh, Z. Shah, and G. Jang, "Line-break prediction of hanmun text using recurrent neural networks," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 720–724, Oct 2017.

[16] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

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

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.

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

[21] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, (Washington, DC, USA), pp. 1653–1660, IEEE Computer Society, 2014.

[22] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4929–4937, IEEE, June 2016.

[23] S. BÄk, E. Corvee, F. Bremond, and M. Thonnat, "Multiple-shot human reidentification by mean riemannian covariance grid," in *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 179–184, 2011.

[24] A. R. Mier y TerÃ¡n, L. Lacassagne, A. H. Zahraee, and M. GouiffÃ s, "Real-time covariance tracking algorithm for embedded systems," in *2013 Conference on Design and Architectures for Signal and Image Processing*, pp. 104–111, 2013.

[25] I. O. de Oliveira and J. L. de Sousa Pio, "Object reidentification in multiple cameras system," in *2009 Fourth International Conference on Embedded and Multimedia Computing*, pp. 1–8, 2009.

[26] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof, "Person re-identification by descriptive and discriminative classification," in *Image Analysis* (A. Heyden and F. Kahl, eds.), (Berlin, Heidelberg), pp. 91–102, Springer Berlin Heidelberg, 2011.

[27] F. Fleuret, H. Ben Shitrit, and P. Fua, "Re-identification for improved people tracking," *Person Re-Identification*, pp. 309–330, 2014.

[28] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, "Alignedreid: Surpassing human-level performance in person re-identification," *arXiv preprint arXiv:1711.08184*, 2017.

[29] X. Zhu, X. Jing, X. You, X. Zhang, and T. Zhang, "Video-based person re-identification by simultaneously learning intra-video and inter-video distance metrics," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5683–5695, 2018.

[30] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang, "Joint detection and identification feature learning for person search," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3376–3385, 2017.

[31] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang, "Person re-identification with deep similarity-guided graph neural network," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[32] S. Li, S. Bak, P. Carr, and X. Wang, "Diversity regularized spatiotemporal attention for video-based person re-identification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 369–378, 2018.

[33] W. Zhang, B. Ma, K. Liu, and R. Huang, "Video-based pedestrian re-identification by adaptive spatio-temporal appearance model," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 2042–2054, 2017.

[34] J. Dai, P. Zhang, D. Wang, H. Lu, and H. Wang, "Video person re-identification by temporal residual learning," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1366–1377, 2019.

[35] M. Li, X. Zhu, and S. Gong, "Unsupervised tracklet person re-identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[36] D. Reid, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[37] G. L. Masala, B. Golosio, M. Tistarelli, and E. Grosso, "2d recurrent neural networks for robust visual tracking of non-rigid bodies," in *Engineering Applications of Neural Networks - 17th International Conference, EANN 2016, Aberdeen, UK, September 2-5, 2016, Proceedings*, pp. 18–34, 2016.

[38] J. Dequaire, P. Ondruska, D. Rao, D. Z. Wang, and I. Posner, "Deep tracking in the wild: End-to-end tracking using recurrent neural networks," *I. J. Robotics Res.*, vol. 37, no. 4-5, pp. 492–512, 2018.

[39] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 4846–4855, 2017.

[40] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially supervised recurrent convolutional neural networks for visual object tracking," *CoRR*, vol. abs/1607.05781, 2016.