Intelligent Camera Selections in a Camera Network

By

Anil Sharma
MT12063

Under the supervision of Dr. Saket Anand and Dr. Sanjit Krishnan Kaul

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

July, 2022

Intelligent Camera Selections in a Camera Network

By

Anil Sharma
MT12063

A Thesis

submitted in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

July, 2022

Certificate

This is to certify that the thesis titled *Intelligent Camera Selections in a Camera Network* being submitted by *Anil Sharma* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Doctor of Philosophy, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

July, 2022

Dr. Saket Anand

Dr. Sanjit Krishnan Kaul

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

Abstract

Surveillance camera networks are a useful monitoring infrastructure that can be used for various visual analytics applications, where high-level inferences and predictions could be made based on target tracking across the network. Most multi-camera tracking works focus on re-identification problems and trajectory association problems. However, as camera networks grow in size, the volume of data generated is humongous, and scalable processing of this data is imperative for deploying practical solutions. One common task in a camera network is inter-camera tracking (ICT). In ICT, once a target leaves a camera's field of view, it needs to be re-identified in the new camera feed after the transition. However, the relative distances between cameras and indeterminate target-transition time make the re-identification (Re-ID) based ICT problem very challenging. With increase in number of Re-ID queries, there is an increase in false alarms as well as the computation time, which can adversely affect tracking performance. In this dissertation, we ask the crucial question of whether to make a Re-ID query or not and selecting which camera to query at each time-step. Our formulation of this decision making problem naturally fits a Reinforcement Learning (RL) framework, which is then solved using a DQN approach for making camera selection decisions. We show that an RL policy reduces unnecessary Re-ID queries and therefore the false alarms, scales well to larger camera networks, and is target-agnostic. We learn a policy for camera selections directly from the data and it has no reliance on the camera network topology.

We further demonstrate that by using learned state representations, as opposed to hand-crafted state variables, we are able to achieve state-of-the-art results on camera selection, while reducing the training time for the RL policy. And we train the DQN in a semi-supervised way to reduce dependence on per frame reward. We use accumulated discounted reward to train DQN and show that it achieves comparable performance to DQN when trained with per frame reward. We demonstrate that using camera selections in a camera network

benefits applications such as multi-target multi-camera (MTMC) tracking and multi-camera target forecasting (MCTF). We report our results on four datasets: NLPR_MCT, DukeMTMC, CityFlow dataset, and WNMF dataset.

Along with it, we also propose a new experience replay method for DQN to work with imbalance replay buffer. We analyze why DQN fails to learn a better policy for longer transitions of a target in a camera network and show the limitations of DQN when the replay buffer is imbalanced with the most frequent action. In this direction, we propose modification to existing replay method by using the reward received for the each experience. We show that the proposed experience replay method (named SER) helps to create a diverse mini-batch to train DQN and achieves better performance than existing experience replay methods.

■

Acknowledgements

My PhD journey had many ups and downs and throughout this journey, many people supported my efforts and motivated me to do good research. First of all, I acknowledge Infosys Center for Artificial Intelligence (CAI) at IIIT-Delhi and the UGC Junior Research Fellowship (JRF) by Government of India which provided me consistent funding support without which it would have been difficult to complete this research.

I would like to express my sincere gratitude towards my supervisors, Prof. Saket Anand and and Prof. Sanjit K. Kaul for their constant support throughout my PhD. I would also like to thank them for making my life easy at IIIT Delhi as they have always encouraged me during this journey. I would also like to thank my PhD monitoring committee for their consistent feedback having, formerly, Prof Chetan Arora, Prof P.B. Sujit and later, Prof A V subramanyam and Prof Pravesh Biyani.

I would also like to thank several other people who kept me motivated in this journey especially people whom I met during conferences and workshops. First of all, Prof. Shivaram Kalyanakrishnan from IIT-Bombay whom I met in IIT-Kanpur during a workshop and he suggested to use contextual bandits for my work. Only after this, we started working on reinforcement learning and published a few papers. Then, Dr. Parthasarathy Sriram from Nvidia whom I met at IJCAI-2018 and he was very excited seeing my work. He showed what Nvidia is working on for multi-camera vehicle tracking. That discussion was a source of motivation for me because of which I kept working on the same problem to solve a few more challenges.

I would also like to extend my deepest gratitude to my peers and lab mates, Lokender, Manoj, Milan, Haroon, Pravin, and Tanya with whom I have made several technical discussions on my research work and gave good feedback. I am also thankful to Ankita, Anupriya, Sharat, Mayank, Parikshit, Alvika for

their consistent support during my PhD. I would also like to thank Krishan, Rahul, Omkar, Kaushik for being with me in exploring several parts of North and east India. I would also like to thank Dr Arun Balaji at IIIT-Delhi for having a great discussion on one of my problem and as a result of this discussion we had an extended abstract in AAMAS-2018. I would also like to thank my other friends whom I met during conferences and made several talks post conference as well especially, Neeraj, Shalini, Joseph, Sai, Phaniraj, Shakti, Laxmi Narayana. Having friends like them is a blessing, they were always available for making a technical discussion.

Last but not the least, I express my deepest appreciation for my parents and my sisters for their unconditional support and belief in me. They have supported me in all aspects because of which I was able to finish my Ph.D. stress free. I am also thankful to my friends at Iskcon especially Sadbhuj Gaur and Arindam (Ananda Keshva) for showing me a different perspective of life.

Anil Sharma

Publications directly related to the thesis

Refereed Journals

- **Anil Sharma**, Saket Anand, and Sanjit K. Kaul. "Intelligent querying for target tracking in camera networks using deep q-learning with n-step bootstrapping." Image and Vision Computing 103 (2020): 104022.

Refereed Conferences

- **Anil Sharma**, Saket Anand, and Sanjit K. Kaul. "Intelligent Camera Selection Decisions for Target Tracking in a Camera Network." *to appear* In IEEE Winter Conference on Applications of Computer Vision (WACV) 2022.
- **Anil Sharma**, Mayank K. Pal, Saket Anand, and Sanjit K. Kaul. "Stratified Sampling Based Experience Replay for Efficient Camera Selection Decisions." In 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM), pp. 144-151. IEEE, 2020.
- **Anil Sharma**, Saket Anand, and Sanjit K. Kaul. "Reinforcement learning based querying in camera networks for efficient target tracking." In Proceedings of the International Conference on Automated Planning and Scheduling, vol. 29, pp. 555-563. 2019.

Other Publications during Ph.D.

- **Anil Sharma**, and Sanjit Kaul. "Two-stage supervised learning-based method to detect screams and cries in urban environments." IEEE/ACM Transactions on Audio, Speech, and Language Processing 24, no. 2 (2015): 290-299.
- **Anil Sharma**, and Arun Balaji Buduru. "Foresee: Attentive future projections of chaotic road environments." In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, pp. 2073-2075. 2018.
- Mayank K. Pal, Rupali Bhati, **Anil Sharma**, Sanjit K. Kaul, Saket Anand, and P. B. Sujit. "A reinforcement learning approach to jointly adapt vehicular communications and planning for optimized driving." In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 3287-3293. IEEE, 2018.

Contents

Abstract i

Acknowledgements iii

Publications v

List of Tables xi

List of Figures xv

List of Acronyms xix

1 Introduction 1

1.1 Tracking in a Camera Network 1

1.2 Camera Selections for Querying in a Camera Network 4

1.3 Summary of Contributions 6

1.3.1 Querying in a Camera Network for Efficient Target Tracking 7

1.3.2 Stratified Sampling Based Experience Replay 8

1.3.3 State Representation Learning Based Camera Selection Decisions 9

1.4 Challenges and Assumptions 10

1.5 Dissertation Organization 11

2 Literature Survey **14**

2.1 Tracking in a Camera Network 14

2.2 Deep Reinforcement Learning for Visual Tracking 17

3 Querying in a Camera Network for Efficient Target Tracking **18**

3.1 Introduction 19

3.2 Proposed Methodology 23

3.2.1 Problem Formulation 23

3.2.2 System Architecture 28

3.3 Experiments and Results 28

3.3.1 Dataset and Evaluation Metric 29

3.3.2 Experiments 32

3.4 Discussion 40

4 Intelligent Querying in a Camera Network Using Deep Q-learning with n-step Bootstrapping **41**

4.1 Introduction 42

4.2 Proposed Methodology 46

4.2.1 System Overview 46

4.2.2 Markov Decision Process and Q-learning 47

4.2.3 Camera Selection Decisions using Deep-Q Network . . 54

4.3 Evaluation and Results 59

4.3.1 Dataset and Evaluation Metric 59

4.3.2 Camera Selection Performance of the Learned Policy . . 61

4.3.3 Impact of Camera Selection Decisions on Target Tracking in Camera Networks 68
4.3.4 Comparison with State-of-the-art methods 70
4.4 Limitations 71
4.5 Discussion 72

5 Stratified Sampling Based Experience Replay 73
5.1 Introduction and Motivation 74
5.2 Background and Related Work 77
5.3 Formulation as an MDP 79
5.4 Proposed Approach 82
5.4.1 Problem Statement 82
5.4.2 Proposed Experience Replay Approach 84
5.5 Experiments and Results 87
5.5.1 Experimental Setup 88
5.5.2 Performance comparison 91
5.5.3 Analysis of Sampled Transitions 93
5.6 Discussion 95

6 State Representation Learning Based Camera Selection Decisions 96
6.1 Introduction 97
6.2 Proposed Method 100
6.2.1 Camera Selection as an MDP 101
6.2.2 System Architecture 103
6.2.3 Camera Selection Policy Model 105
6.3 Results 109
6.3.1 Experimental Setup 110

6.3.2 Camera Selection Decisions 113
6.3.3 Scalable Camera Selection Decisions 118
6.3.4 Benefits of Camera Selection Decision in a Camera Network 119
6.4 Discussion 125

7 Conclusion and Future Work **127**

References **129**

List of Tables

3.1 Details of NLPR-MCT dataset [1], which has four subsets. The table shows number of cameras (#Cameras), duration of the capture, frame rate (FPS) and the number of people (#People) captured in each subset. 29

3.2 Table is showing confusion matrix of the camera selections made by the proposed policy for DB-3. Rows are the ground truth cameras and columns are the cameras polled by the policy. Values are percentages rounded off to third decimal. 30

3.3 Table showing average time taken (in the number of frames) by all targets from camera C_i (row) to camera C_j (column). The values (g, p) are ground truth (g) time and time taken by the policy (p) to find the target in the next camera. The values are averaged over all targets in the test set of sub-dataset 3. 30

3.4 Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for ICT alone and for both SCT with ICT. . . 32

3.5 Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for ICT alone and for both SCT with ICT. . . 33

3.6 The table is showing average MCTA values (higher is better) for inter-camera tracking (ICT) and both SCT-ICT on the test set of NLPR-MCT dataset. The related approaches are multi-camera multi-target tracking approaches taken from the benchmark dataset [2]. The last 5 rows show the MCTA values for the proposed approach with simulated Re-ID errors from 0% to 20%. 36

4.1 Details of the datasets used for performance evaluation. The table shows the number of cameras (#Cameras), duration of the videos, frame rate (FPS) and the number of people (#People) captured in each dataset. 59

4.2 Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for the case of Inter-Camera Tracking (ICT). . 61

4.3 Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for the case of both ICT and SCT together. . . 62

4.4 Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for DukeMTMC dataset for both ICT alone and ICT-SCT together. The Gaussian approach is not defined for SCT+ICT case. In the table, OM signifies Out-of-Memory error. 62

4.5 The table is showing average MCTA values (higher is better) for inter-camera tracking (ICT) on the test set of NLPR-MCT dataset. The related approaches are multi-camera multi-target tracking approaches taken from the benchmark dataset [2]. The last 10 rows show the MCTA values for the proposed approach with simulated re-identification errors from 0% to 20% for both Exact RL and Deep RL implementations. 67

4.6 The table is showing average MCTA values (higher is better) for SCT and ICT together case on the test set of NLPR-MCT dataset. The related approaches are multi-camera multi-target tracking approaches taken from the benchmark dataset [2]. The last 10 rows show the MCTA values for the proposed approach with simulated re-identification errors from 0% to 20% for both Exact RL and deep RL implementation. 67

4.7 The table is showing average MCTA values (higher is better) for both SCT+ICT and ICT alone case on the DukeMTMC dataset. OM signifies Out-of-Memory error. There are no related approaches that define the tracking performance on DukeMTMC dataset using MCTA scores. 69

5.1 Table is showing camera selection accuracy (A), precision (P) and recall (R) for the different methods on various camera network datasets. SER is our proposed experience replay method. . 90

5.2 Table is showing camera selection accuracy (A), precision (P) and recall (R) for the different methods on various camera network datasets. SER is our proposed experience replay method. . 91

6.1 Details of the datasets used for training and performance evaluation. The table shows the number of cameras (#Cameras), duration of the videos, frame rate (FPS), the number of targets (#Target) captured in each dataset. 111

6.2 Camera Selection performance of our proposed method and its comparison with state-of-the-art approaches for NLPR-set1,2,3. The best results are shown in bold and second-best results are italicized. 112

6.3 Camera Selection performance of our proposed method and its comparison with state-of-the-art approaches on NLPR-Set4 and DukeMTMC dataset. The best results are shown in bold and second-best results are italicized. 113

6.4 Testing loss on different datasets with different LSTM size. The corresponding epoch number is indicated in brackets. 114

6.5 Percentage Camera Handovers (PCH) (higher is better) on NLPR-Set4 when trained without AE and with AE. AE(same) represents AE is trained on same dataset, AE (N) represents that AE is trained on a bigger dataset with sequence length N. 115

6.6 Camera selection performance for semi-supervised training on NLPR Set-4. 118

6.7 The camera selection performance on two scenarios of the CityFlow dataset. 119

6.8 Average MCTA values (higher is better) for ICT alone case on NLPR-MCT and DukeMTMC dataset. The results are separated based on the type of association method. *Self* means a method uses its own association, *GT* represents ground truth, and *Re-ID* signifies that a Re-ID method is used for association. We used ABDNet [3] for Re-ID. 120

6.9 Average MCTA values (higher is better) for both SCT-ICT case on NLPR-MCT and DukeMTMC dataset. The results are separated based on the type of association method. *Self* means a method uses its own association, *GT* represents ground truth, and *Re-ID* signifies that a Re-ID method is used for association. We used ABDNet [3] for Re-ID. 120

6.10 The camera selection performance on WNMF dataset. The baseline methods are taken from dataset baselines [4] and LSTM (Cam. Sel.) is a camera selection based baseline. 122

List of Figures

1.1 Summary of contributions in this thesis. 7

1.2 Top view of DukeMTMC dataset [5]. The figure shows the top view of the camera network with field-of-view of all the eight cameras. The camera network is deployed in Duke university campus. 10

3.1 Camera topology of NLPR-MCT dataset-4 [2]. The figure shows the trajectories of person 5 and 6 across different cameras. The camera network is deployed in a parking area and all cameras have non-overlapping view. 20

3.2 The cells in the grid capture the spatial location of the target. The image is first discretized in 8×8 grid, and all cells are numbered sequentially in row-major order. The cell number on the target position is used as the spatial location of the target named r_t at time t. One cell value among all r_t is used in the state vector. 26

3.3 The proposed architecture using reinforcement learning. The architecture shows two blocks, block Q and block *presence*. Block Q learns a policy to select a new camera using current state and block *presence* verifies whether the target is present in the camera frame chosen. 27

3.4 The figure shows the transitions for 4 targets in the testing set of dataset-3. GT is the sequence of cameras in ground-truth, Sel is the sequence of cameras selected/polled by the policy. Horizontal axis is the time. White color is the time when the target is transitioning between cameras and colorbar depicts the camera numbers in the plot. 33

3.5 The figure shows the learned topology for set 4 (5 cameras) and 3 (4 cameras). A black arrow indicates the correct prediction and red arrow indicate a false positive. 35

3.6 Boxplot of number of frames polled (metric F, see equation 3.9) on two datasets of NLPR dataset for our proposed policy and other baseline approaches. 36

4.1 The neural network model that learns the state-action values using Q-learning. The model learns a policy that makes the camera selection decisions. This is the implementation of the Q block of the architecture shown in Figure 3.3 on page 27 in Chapter 3. 55

4.2 Analysis of training strategy. First, shows the varying epsilon value during training on NLPR DB-3. Second, the running reward during the training on NLPR DB-4. 58

4.3 The figure shows confusion matrix of the camera selections made by the proposed policy for DukeMTMC dataset. Rows are the ground truth cameras (GT) and columns are the cameras polled by the policy. Values are percentages rounded off to third decimal. 66

4.4 The figure shows the transitions for 7 targets in the testing set of dataset-3. On y-axis, GT is the sequence of cameras in ground-truth, Sel is the sequence of cameras polled by the policy. Horizontal axis is the time. White color is the length of the transition during camera handovers and colorbar depicts the camera numbers in the plot. 66

4.5 Number of frames polled (F, equation 3.9) on DukeMTMC dataset for our deep RL based policy and its comparison with other baseline approaches. 70

5.1 Example camera transitions for 3 targets P_1, P_2, and P_3. 76

5.2 The figure shows the training time on x-axis and the number of samples of each transition type on y-axis. The transition type is shown in the legend. The replay memory is dominated by the most frequent action $C_{\times}$. 83

5.3 Overview of the proposed experience replay method. R_f, R_{-}, and R_{+} are the different replay memories to store frequent, negative reward, and positive reward transitions respectively. 86

5.4 Qualitative performance of different ER methods. In the figure, each color represents a specific camera FOV and time gap between colors show the transition time of the camera handover. 89

5.5 Percentage Camera handovers (PCH) (higher is better) captured by different ER methods. SER is largest for all datasets. 90

5.6 Number of spurious frames polled (F in equation 3.9) on a NLPR_-Set4 dataset and b DukeMTMC dataset. The figure also shows the comparison of our proposed policy and other baseline approaches 92

5.7 Figure showing reward diversity in the sampled minibatch of different ER methods for Set-3 for PER, ER-Uniform, and ERO. Results for PER on Set-4 are also shown. 94

6.1 A) The architecture of the LSTM based autoencoder that is used to encode the action history in a fixed length latent representation (Z). B) The DQN architecture used to learn the camera selection policy. The neural network model that learns the policy takes as input the different state variables and the action history (h_t) encoded using the LSTM based Autoencoder (E-Encoder, D-Decoder). 100

6.2 Figure shows the modification in the reward function (equation 6.1) for semi-supervised training. 110

6.3 Figure shows the qualitative performance of the proposed camera selection method for two targets. 113

6.4 The re-identification calls made by different methods on DukeMTMC dataset. 116

6.5 Figure compares the Percentage Camera Handover (PCH) of our method with the state-of-the-art method. 118

6.6 Ground Truth topology and predicted topology of the DukeMTMC dataset. Both axes show the camera index and color intensity represents the number of transitions in a $c_i - c_j$ transition. . . . 119

6.7 Figure shows the difference in the transition time captured of multiple tracks/targets of WNMF dataset. 125

6.8 Figure shows the difference in the transition time captured of multiple tracks/targets of WNMF dataset. 126

List of Acronyms

Re-ID Re-identification

SCT Single Camera Tracking

ICT Inter Camera Tracking

FOV Field of View

FPS Frames Per Second

MDP Markov Decision Process

RL Reinforcement Learning

DukeMTMC Duke Multi Target Multi Camera Dataset

NLPR-MCT NLPR Multi Camera Tracking Dataset

DQN Deep Q Network

SER Stratified Sampling Based Experience Replay

LSTM Long Short Term Memory

AE Auto Encoder

MCTA Multi Camera Tracking Accuracy

GT Ground Truth

TD Temporal Difference

A Accuracy

P Precision

R Recall

F1 F1 score

F Spurious Frames Queried

IDP ID Precision

IDR ID Recall

IDF1 ID F1 Score

ER Experience Replay

PPO Proximal Policy Optimization

PER Prioritized Experience Replay

ERO Experience Replay Optimization

SRL State Representation Learning

MTMC Multi Target Multi Camera Tracking

MCTF Multi Camera Tracking Forecasting

WNMF Warwick-NTU Multi-camera Forecasting database

MSE Mean Square Error

IOU Intersection Over Union

PCH Percentage Camera Handover

Chapter 1

Introduction

1.1 Tracking in a Camera Network

Camera networks have emerged as a preferred sensing infrastructure for monitoring and surveillance of public spaces. Camera networks are used for various practical applications like assisted living facilities, environmental monitoring, monitoring of vehicles from cameras on road intersections, etc. These applications are driven by state-of-the-art visual detection, tracking and re-identification techniques that are robust to lighting variations, background clutter, occlusions, and non-overlapping fields of view of cameras in the network. In order to track the target in multiple cameras, the camera network is queried several times to re-identify the target during the duration of target's transition from source camera to destination camera. A false alarm from wrong re-identification (Re-ID) will severely degrade the performance and hence it is crucial to reduce re-identification queries for performance to track a particular target. Large number of Re-ID queries will also result in large computation time. This compu-

tational challenge is exacerbated by the deluge of video data generated from a network of cameras, making it challenging to implement these applications at scale. The number of cameras at an airport, railway station, malls, etc. has rapidly increased, which makes automated tracking an essential task for visual analytics. The camera networks generate an enormous amount of video data which makes it difficult to process all video frames in real-time. Finally, in many scenarios, the camera network topology is not known, and the tracking algorithm should be able to track the target in the absence of this knowledge. In this thesis, we look into camera selections which selects a camera where the target is likely to be present in the camera network. We will show that the camera selections improve tracking performance by reducing candidates for Re-ID query and also benefits in the computational time.

Re-identification (Re-ID) and data-association are conventional ways [5, 6] used to associate individual tracklets from different cameras to form the multi-camera trajectory of a particular target. These methods doesn't incorporate the indeterminate and unknown transition time of the target. Methods[7, 8, 9] have shown the tracking performance by modeling the transition time using static models such as Gaussian distribution and Parzen window. Many of these approaches for multi-camera target tracking employ a two-step framework [7, 1, 5, 10]. First, SCT (Single-Camera Tracking) to find the target's trajectory within each camera. Second, ICT (Inter-Camera Tracking), to associate the SCT trajectories corresponding to the same identity across camera after the target has transitioned from one camera's FOV to another.

The Re-ID or association based tracking methods rely excessively on the performance of Re-ID. A small false alarm in Re-ID can create huge performance failures in the target tracking. For example, if we consider a conservative case of a camera network with 5 cameras and only 3 persons per camera. In this case, one minute of 10 FPS of this dataset generates around 9000 person bounding boxes. To look for the trajectory of the target, a basic Re-ID based tracking method would query 9000 times to the camera network using a re-identification method and 1% false alarm in the re-identification method can severely fail to generate the target's trajectory. Along with the performance failures, the Re-ID is computationally expensive in terms of all-pair matching. Therefore, there are a few questions that should be answered for efficient target tracking. For example, do we need to perform all-pair matching (or matching to the linked cameras)? Do we need to match a candidate template all times when the target leaves a camera FOV and transitioning to another camera's view? Can we scale the same solution easily for tracking targets in a large campus like IIIT-Delhi where there are hundreds of cameras and thousands of people roaming around?

The answer to these questions is obviously computationally expensive using existing methods and these questions are crucial to say how efficiently can we track targets in the camera network. Therefore, we need an intelligent method that relies less on querying the camera network all times. To achieve this, we propose camera selections in a camera network where an intelligent policy can keep track of the camera where the target is expected to be present. The objective of this thesis is to propose an efficient approach to use the multiple video

streams of a camera network. In this thesis, we investigate intelligent camera selection and focus on tackling the problem of camera-handovers, as we scale to larger camera networks.

1.2 Camera Selections for Querying in a Camera Network

One common task in a camera network is inter-camera tracking (ICT). In ICT, once a target leaves a camera's field of view, it needs to be re-identified in the new camera feed after the transition. However, the relative distances between cameras and indeterminate target-transition time make the re-identification (Re-ID) based ICT problem very challenging. With increase in number of Re-ID queries, there is an increase in false alarms as well as the computation time, which can adversely affect tracking performance. Therefore, to make the querying efficient, we can ask a crucial question whether to make a Re-ID query or not. If a query needs to be made then in which camera the target is likely to be present? To achieve this, we formulate this decision making problem as a Markov Decision Process (MDP) and use Q-learning, a famous reinforcement learning based method, to learn a policy for making camera selection decisions. We will show that an RL policy reduces unnecessary Re-ID queries and therefore the false alarms, scales well to larger camera networks, and is target-agnostic. We will further demonstrate that by using a state representation based method, as opposed to hand-crafted state variables, we can learn a policy faster. We will also show that using reinforcement learning can help to

reduce the dependence on frame level annotation for training the Q-learning agent . We will show that our reward structure to the learning agent reduces dependence on the frame level annotations and helps to train the policy with limited supervised data. Additionally, the RL method helps us to learn a policy without any information from the camera network (such as camera network topology) and a policy is learned directly from the data. Along with these, the RL approach effectively makes camera selection in indefinite transition times and avoid modeling of the transition time using any static distribution.

It should be noted that we do not rely on the camera network topology and we solve a more general problem where such an information is rarely available. It can be noted that this is a static information and one-time overhead to obtain this information. We will show that having this information is not sufficient for making camera selection decisions or target tracking using conventional methods. The more challenging aspect is the target transition time which is indefinite and not known in advance as it depends on various factors such as target's speed, its destination, congestion, etc. We will show that modeling it as a static distribution does not provide better performance whereas we use a state vector to capture the target's movement automatically. We use the target's last seen location, the time information, and the subsequent selected camera history. Using these variables in the state, we are able to track the transition time of the target effectively. Along with it, we will also show that our learned policy implicitly learns the camera network topology and hence we do not rely on this information.

The reinforcement learning based policy is an ideal solution to track targets in the camera network as it relies less on the re-identification. To clarify the objective of this thesis, we don't propose any changes to the re-identification whereas our framework can make use of any public re-identification based method once a successful query is made to the camera network. We will show through various experiments that camera selections are very crucial and improves the tracking performance as compared to using only a re-identification based approach. We will use a pre-trained re-identification method on DukeMTMC dataset [5] and will use the same model for other datasets as well. Using our proposed method, we make only one Re-ID query or no-query at a time to the camera network for tracking one target whereas the existing two-step method will make several Re-ID queries (equal to the number of cameras multiplied by the number of targets to re-identify the target in the camera network after the transition). This makes our method highly efficient in terms of number of queries made to the camera network. Our experiments quantify that this is more than 100x improvement in the number of queries on the DukeMTMC dataset. Hence, the RL policy helps to scale the target tracking on larger camera networks.

1.3 Summary of Contributions

In this section, we have provided a detailed summary of contributions. We have highlighted a brief summary of the contributions in Figure 1.1.

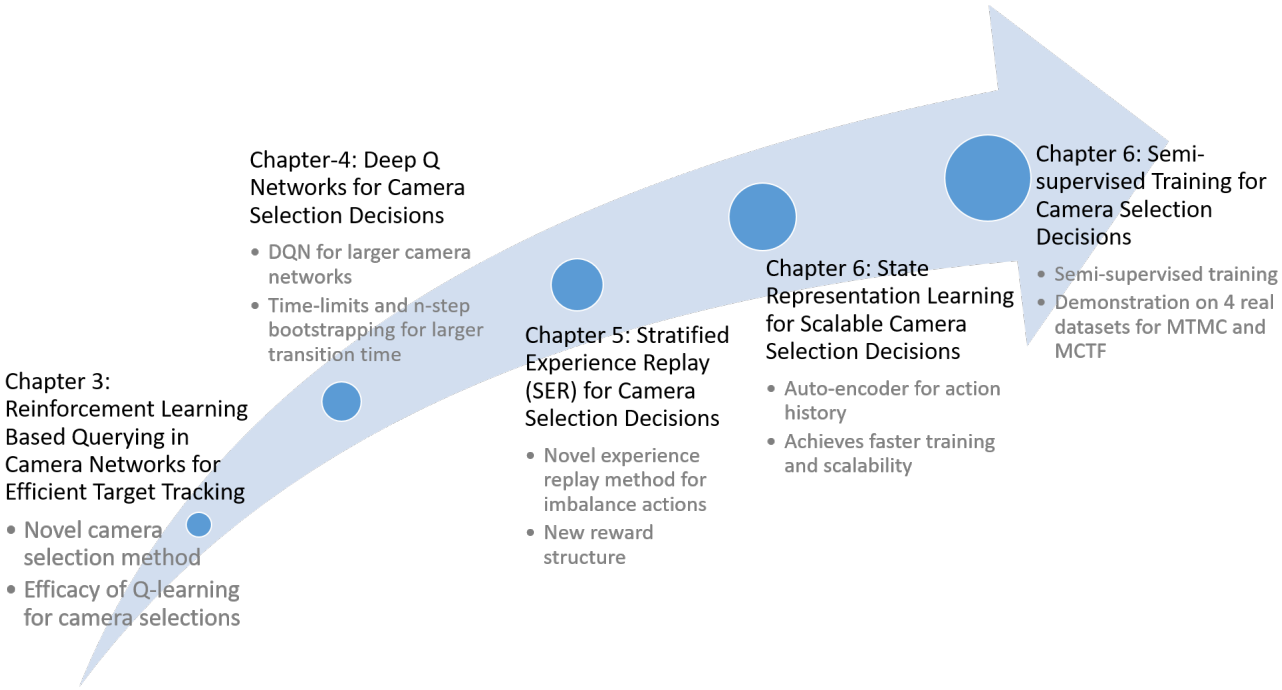


Figure 1.1: Summary of contributions in this thesis.

1.3.1 Querying in a Camera Network for Efficient Target Tracking

Target tracking in a camera network was previously looked from various aspects like using features of target's appearance, context information, modeling the transition time of the target, etc. These methods work in a two step manner, first to identify the trajectory of the target in a single camera, second to associate these single camera tracks to identify the multi-camera trajectory of the target. These methods perform all-pair matching of target position which is highly computationally expensive. This all-pair matching excessively relies on the re-identification which substantially degrades the tracking performance. In this dissertation, we will first show that camera selections are essential for efficient target tracking in a camera network. In this chapter, our focus is to show the efficacy of camera selection decisions for efficient querying in the camera

network. We model the camera selection approach using reinforcement learning and learn a policy that picks an action of querying one of the candidate cameras or not querying any camera. The learned RL policy implicitly discovers the camera network topology, with our only assumption about the network being that all cameras are static. In our experiments, we evaluate our approach with real data (NLPR-MCT dataset [2]) where the targets are restricted to pedestrians, and compare them to the state-of-the-art. In Chapter 4, we will then scale the camera selection approach using Deep Q-learning (or Deep Q network or DQN), a neural based implementation of the Q-learning algorithm. Using time-limits[11] and n-step bootstrapping, we will show that our method works effectively for making camera selection decisions in a larger camera network and handles indefinite transition times, while still maintaining the MDP formulation and learn the policy using DQN.

1.3.2 Stratified Sampling Based Experience Replay

The camera selection in a camera network are highly effective in reducing reliance on the re-identification. It improves the performance and reduces the number of queries to the camera network. We point out that the number of camera handovers are very less than the the time instances when the target is transitioning between cameras. In Chapter 5 of this dissertation, we will show that the camera selection approach is highly imbalanced towards one action of deciding when to query a camera. This is an important observation with respect to training a deep RL model, which requires appropriately handling of imbalanced

state transitions during experience replay. We will analyze why traditional deep Q-learning fails to learn a good policy for larger transition times and propose a modification to the experience replay method for handling the imbalance in exploration of actions. Our proposed approach, referred to as Stratified Experience Replay (SER) resolves this challenge by sampling in the imbalanced replay memory created by the different episodic runs of the agent-environment interaction. We will show on various datasets that SER with Q-learning learns a better policy for camera selections than existing experience replay methods.

1.3.3 State Representation Learning Based Camera Selection Decisions

We observe that in the absence of knowledge of the camera topology, the camera history is an important state variable. It holds information about the sequence of previously queried cameras, which influences the decision of which camera to select for the next query. For larger camera networks, retaining longer history of camera selection is necessary to make well-informed camera selection decisions but retaining that as one-hot vector is computationally expansive. In this work, we argue that hand-crafted state variables may not be representative enough and hinder the scalability of such an approach. Therefore, we instead propose a state representation learning [12] based approach and modify the state-vector accordingly. A representation helps to learn the variations in the environment in a low dimension vector. Our final state vector leverages an LSTM-based autoencoder (AE) to summarize the camera history of Re-ID queries. We further show various advantages of using a learned state representation, including

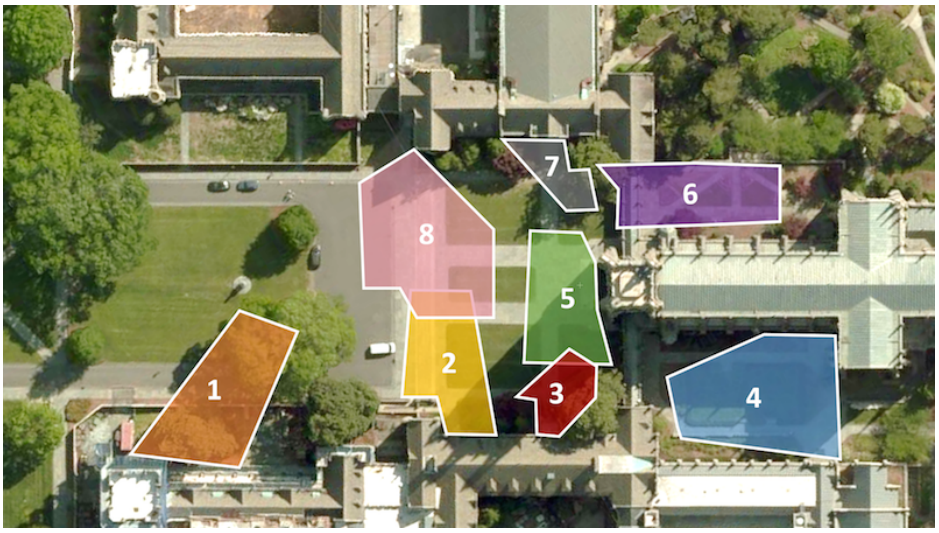


Figure 1.2: Top view of DukeMTMC dataset [5]. The figure shows the top view of the camera network with field-of-view of all the eight cameras. The camera network is deployed in Duke university campus.

generalization across camera-network datasets, accommodating a generic DQN architecture across datasets and most importantly reduced training speeds. We will show that the AE trained once on a larger dataset works for all smaller datasets as well. In addition to this, we leverage the reward structure to train it in a semi-supervised manner. We will show that providing an annotation every fifth frame achieves comparable performance with all frame annotation. This is a good step toward thinking of an approach which does not fully rely on all frame annotations.

1.4 Challenges and Assumptions

To understand the challenges of tracking a target in a camera network, let's consider the camera network shown in Figure 1.2. The figure shows the top view of the cameras installed in DukeMTMC dataset. There are 8 cameras and the colored polygons show the FOV of these cameras. The one of the common chal-

lenge is the unconstrained path and unknown camera network topology. Many of the datasets do not have the network topology information and hence the tracking algorithm should be able to track the targets even when this information is not available. Another challenge is the non-overlapping field of views (FOVs) and hence there are blind spots between cameras where the target is not visible in any of the cameras. The target transition time is indefinite and hence the tracking algorithm should be vigilant of when the target will re-appear in the camera network to make the tracking task efficient. The tracking algorithm should also track the target when it is occluded in the single camera view. Along with these challenges, the camera networks generate huge amount of data and hence the tracking algorithm should be efficient enough to track the target.

We make an assumption that the camera network doesn't change during or after training which means that no camera is removed or added to the camera network. We also assume that all camera are static. Apart from it, we don't use any information of the camera network. One common limitation of our method is that the learned policy cannot adapt to the changes in the camera network and a separate policy needs to be trained. Despite this limitation, we showed that our methods perform better on several datasets and achieve state-of-the-art performance.

1.5 Dissertation Organization

The rest of this dissertation is organized into following chapters:

- Chapter 2 provides a brief survey of the related methods on target tracking algorithms in a camera network. The chapter will also describe the deep reinforcement learning based literature for visual tracking. The literature survey describes both deep learning and earlier methods in computer vision that use target features and context information for tracking targets.

- Chapter 3 proposes camera selection decisions for querying in a camera network. In this chapter, we describe the camera selection method and provide details to learn a policy using tabular Q-learning method . We will show that our method achieves better performance than several baseline methods.

- Chapter 4 provides the formulation of camera selection decisions using a Markov Decision Process (MDP). In this chapter, we will provide details to learn a policy using deep Q-learning method for larger camera networks.

- In Chapter 5, we will show the limitations of deep Q-learning in learning an optimal policy for larger camera networks. We will then propose a new experience replay method named *Stratified Experience Replay* to sample a diverse minibatch from the replay buffer.

- In Chapter 6 , we show the efficacy of state-representation learning to learn a policy faster by encoding the action history in a small latent representation. Using LSTM based encoder helps to reduce the training time of the policy and achieves better camera selection performance. We will show that our method reduces reliance on frame level annotation for learning a

camera selection policy.

- In Chapter 7, we conclude this thesis and provide future directions for extending our work for camera selection and target tracking in a camera network.

■

Chapter 2

Literature Survey

2.1 Tracking in a Camera Network

Multi-camera tracking is looked from various viewpoint in both overlapping and non-overlapping cameras. Initially, 3D coordinates of the target's location were used for tracking targets in multiple cameras . Works such as [13, 14, 15, 16, 17] assumed overlapping camera field-of-views (FOVs). These require camera calibration and knowledge of camera network topology to obtain the 3D coordinates. Few other works use network flow graph [14], Kalman filter [15] on the 3D coordinates. Tracking in overlapping cameras is relatively simple because non-overlapping cameras require to handle the blind spot areas between cameras.

To resolve camera handovers in non-overlapping FOVs, a few initial works have created a social group model [18] to associate target tracklets, affinity model [19] of target's appearance for inter-camera association. Other works

formulate various data association methods [20, 21, 22] to resolve camera handovers and use graph [18, 1, 23, 24, 25, 26] based approaches for inter-camera tracking. Spatio-temporal contextual information [10], clique based methods [27, 28], part based model [29, 30] are also a few other common approaches. Many work perform pairwise matching [31, 32, 33, 34, 35, 36] of the templates to form trajectories. Template re-identification [37, 36] approaches are leading for matching target's template with other candidate templates. To resolve the all pair matching issues, multiple method [5, 38] associate only time consecutive templates to reduce computational complexity [5]. In this regard, works [8, 20] use the travel time of the target to estimate the transition time of the camera handover. Works [7] have estimated a transition time distribution using a Gaussian distribution. In comparison to these works, to handle the indeterminate transition times, we propose a reinforcement learning based policy that selects a camera index where the target is expected to reappear after handover. In our experiments, we will show that using such a policy reduces the number of search queries made to the camera network.

To tackle the illumination variation, many works [39, 14, 22] have proposed features using color [14, 8, 19, 40, 25], texture [22, 21, 18], color normalization [10], shape [22, 7], brightness transfer [8, 40, 41, 42], and learning based methods [5]. These appearance features are combined with spatio-temporal reasoning [13, 19], graph based methods [1] to perform the association. The appearance features were also used with Bayesian inference [43] by integrating the features of color and size of the target but this is limited to two camera setup

and multi-camera setup using Bayesian inference in [44].

Association based works perform multi-camera target tracking in a unified way. Many recent works perform tracking task in a two step framework. First, they perform single camera tracking (SCT) and second, inter-camera tracking (ICT) to resolve the camera handover separately. Works such as [7, 1, 10, 38, 5, 45] use such a two step framework for tracking in multiple cameras. [7] has proposed an online method to track multiple targets in SCT and ICT separately. They used sophisticated features of human appearance such as head pose and color along with segmentation using change point detection to perform SCT. To perform ICT, they estimate a camera link model using the human appearance features and estimate the travel time using a Gaussian distribution. Other common approaches estimate entry-exit points across cameras [20, 10]. Tracking, searching and ending track approach is used in [46]. Current state-of-the-art in appearance features re-identify a target in different cameras using deep learning based methods [37, 5, 6] including deep feature representation learning, deep metric learning and ranking optimization. [5] learn the correlation features using combinatorial optimization. They have proposed a weighted triplet loss to learn better features of target's appearance. However, their approach tracks a target in an offline fashion and makes a very large number of re-identification queries to the camera network. We use [3] in our work to re-identify a target in any camera. In this thesis, we will show that camera selection decisions are crucial to enable tracking in camera networks and such a policy queries the camera network a very few number of times.

2.2 Deep Reinforcement Learning for Visual Tracking

Many vision problems [47, 48, 49, 50, 51, 52, 53, 54, 55] have been formulated using Markov Decision Process (MDP) [56]. Formulating the tracking problem using MDP is effective because the agent learns to take actions sequentially, which implicitly model the target's motion. In our formulation, we have used MDP for camera selection decision to enable single target tracking in multiple cameras which can easily be extended to tracking multiple targets by simultaneously running multiple policies. Deep-Q learning [51] has shown human level performance in playing Atari games using visual frames. Such methods use one-step reward during the training process, however, n-steps reward [57] (refer Chapter 7) can help in faster convergence by bootstrapping states for multi-step reward. Time limits [11] in reinforcement learning has shown that randomizing the state vector after a time limit achieves better performance.

Recently, deep reinforcement learning techniques were applied for visual object detection [58] and tracking [48, 59, 60]. These approaches are applied for single object/target tracking in a single camera field-of-view. We have shown [9, 61, 62, 63] that a policy learned using reinforcement learning can intelligently poll cameras to reduce the number of frames required for target's template matching. In our approach, we have used deep-Q learning [51] to learn a policy to poll a camera frame at any time-step to look for the presence of the target.

■

Chapter 3

Querying in a Camera Network for Efficient Target Tracking

Over the past few decades, computer vision research has seen tremendous progress in solving problems like visual object tracking and object Re-ID. However, there is a little attention given to tracking in a camera network and the current research is highly computationally expensive due to its reliance on Re-ID to a great extent which also degrades the tracking performance. Hence, target tracking in a camera network still remains a challenging task. In this chapter, we will show that camera selections are essential for efficient target tracking in a camera network. We will give a brief overview of tracking in a camera network in section 3.1. This is followed by problem formulation, system architecture for camera selections and target tracking in section 3.2. We then describe the dataset, evaluation metric and experimental results in section 3.3 before concluding the chapter in section 3.4. In our experiments, we evaluate our approach with real data (NLPR_MCT dataset [2]) where the targets are re-

stricted to pedestrians, and compare them to the state-of-the-art methods.

3.1 Introduction

Camera networks have emerged as a preferred sensing infrastructure for monitoring and surveillance of public spaces. With advances in visual analytics, we see an increasing number of practical applications like footfall estimation and prediction, crowd and traffic flow analysis, content based retrieval for forensics. These applications are driven by state-of-the-art visual detection, tracking and Re-ID techniques that are robust to lighting variations, background clutter, occlusions, and non-overlapping fields of view of cameras in the network. An example of a camera network and trajectories of different targets is shown in Figure 3.1. State-of-the-art techniques that have shown promise in overcoming these challenges often rely on deep learning architectures that are computationally expensive and have substantial hardware requirements like GPUs to run at an acceptable frame rate. This computational challenge is exacerbated by the deluge of video data generated from a network of cameras, making it challenging to implement these applications at scale.

While many vision techniques have been developed to tackle the problems of tracking and Re-ID under illumination variations, clutter and occlusions, there is a much smaller body of work that addresses the problem of camera selection to efficiently identify target handovers across cameras. Inter-camera handovers are resolved by matching the current target's template with potential target in-

Figure 3.1: Camera topology of NLPR-MCT dataset-4 [2]. The figure shows the trajectories of person 5 and 6 across different cameras. The camera network is deployed in a parking area and all cameras have non-overlapping view.

stances detected in candidate cameras, and is typically handled using visual Re-ID techniques. This problem of correctly resolving handovers is difficult, especially when candidate cameras have non-overlapping FOVs, implying that a target's transition time between two FOVs is non-deterministic and unknown. Ideally, no Re-ID queries should be made during the target's transition period, as a larger number of queries result in an increased chance of false alarms, which can *severely deteriorate* the overall tracking performance. An auxiliary consequence of a higher number of queries is significantly higher computational cost. As the transition time is not deterministic, in order to minimize the false alarms, it is important to devise a camera selection policy that intelligently schedules camera Re-ID queries.

Target tracking in a camera network has been explored extensively in the past using various approaches [64, 10, 7, 1, 18, 19, 22]. Most of these approaches track targets in a two-step framework. First, single camera tracking (SCT) is applied to identify the trajectory of the target within a single camera's FOV. Second, inter-camera tracking (ICT) is performed to resolve handovers by find-

ing associations for the target tracked by SCT. In addition to handovers, ICT is also invoked during SCT for handling periods of occlusion, which may vary significantly depending on the targets' speed and path. ICT requires searching targets in multiple cameras at different time instances, by making repeated Re-ID queries until the target is found and SCT is invoked for the identified cameras. In order to reduce the search space for ICT, most existing techniques limit the set of candidate cameras to be queried by assuming knowledge of the camera network topology [1], while some attempt to model the transition time as a random variable [7, 8].

Recently, [7] proposed a method that discovers a *camera-link* model that identifies candidate cameras for ICT based on appearance based features. They model the inter-camera transition time using a Gaussian distribution, which is used during test-time to generate a sample transition time for which their system waits before initiating Re-ID queries. Not surprisingly, this approach for ICT shows an improvement over exhaustive search and nearest neighbor based search, however, it is not clear if a static distribution of transition time is the best model choice.

In this chapter, we argue that the transition time distribution is conditioned on the target instance and its properties like speed, which itself may be time-varying. Consequently, our proposed ICT approach is a policy for scheduling candidate cameras for Re-ID queries based on the target's most recent SCT trajectory, as well as the history of candidate cameras queried. We model our ICT approach as a reinforcement learning (RL) problem and learn a policy that picks

an action of querying one of the candidate cameras or not querying any camera. The learned RL policy implicitly discovers the camera network topology, with our only assumption about the network being that all cameras are static. Also, We train separately for every dataset with different camera network topology. Since the focus of our work is on the camera selection policy for ICT, our state representation abstracts out the visual appearance based features and only retains spatial information from SCT. In our experiments, we evaluate our approach with real data (NLPR_MCT dataset [2]) where the targets are restricted to pedestrians, and compare them to the state-of-the-art.

The NLPR-MCT records real-world scenarios in four different sub-datasets. This dataset has both indoor and outdoor cameras deployed in a campus building, in parking areas, and along footpaths. We will be using this dataset for training and testing of our proposed approach. We will also compare our results with state of the art methods on this dataset. Figure 3.1 shows a sub-dataset of NLPR-MCT dataset.

In this context, our specific contributions are:

- We propose an intelligent camera selection approach for inter-camera tracking in a camera network. The goal is to learn a policy that schedules the Re-ID query by selecting the next candidate camera where the target is likely to appear.
- We formulate the camera selection as a reinforcement learning problem and learn the policy using Q-learning [57], without any knowledge of the

camera network topology.

- We demonstrate that the camera selection policy queries a very small number of frames by making a small trade-off on the recall values.
- We demonstrate our Q-learning based approach on NLPR-MCT [2] dataset implicitly learns the network topology.

3.2 Proposed Methodology

In this section, we formulate the camera selection approach and present an architecture that integrates camera selections with Re-ID to enable target tracking.

3.2.1 Problem Formulation

Target tracking in a camera network needs to handle inter-camera handovers by resolving associations between the tracked target and potential targets detected across all candidate cameras. This association problem is typically done by visual Re-ID or verification methods. When camera networks have disjoint FOVs, a target may only reappear in another candidate camera after a certain *transition time*, which in turn depends on various factors like inter-camera distance and target speed, where the latter would typically be target-dependent and time-varying. Re-ID queries made at times when a target is unlikely to appear in a candidate camera can lead to unnecessary false associations, deteriorating the tracking performance. To counter this challenge of handling the time-varying nature of inter-camera transitions, we attempt to schedule the Re-ID queries by

intelligently selecting a candidate camera or waiting. Due to its time-varying nature, a befitting model for this problem is a reinforcement learning (RL) based camera selection policy that identifies a candidate camera or decides to wait. Thus the task is to learn the policy $\pi(s_t) = p(a_t|s_t)$ at time t, where s_t is the current state (detailed in next paragraph), a_t is the action taken at time t, which corresponds to selecting one of the N cameras or to wait (by picking a *dummy camera*).

Algorithm 1 Target tracking in a camera network using proposed RL based method. π is camera selection policy. `c,b` are current camera and corresponding bounding box for target location.

procedure TRACK(`c,b`, π)
  `traj` $\leftarrow$ [] ▷ Stores computed trajectory
  `traj.append((c,b))`
  $h \leftarrow$ `ZEROS` ▷ Initialize history with ZEROS
  $\tau \leftarrow$ `ZERO` ▷ Initialize telapse to `ZERO`
  `rt` $\leftarrow$ `getRT(b)` ▷ Discretize the frame to get region of target's location
  $s \leftarrow$ `[c,rt,h,`τ`]` ▷ Concatenate location and history
  while True **do**
    `cprob` $\leftarrow \pi(s)$ ▷ Get distribution using policy
    if all(cprob(:) == cprob(1)) **then**
      c = `randi(length(cprob))`
    else
      c = `argmax` (cprob)
    `b` $\leftarrow$ `get the bounding box location`
    if `b` is not empty **then**
      `rt` $\leftarrow$ `getRT(b)`
      `traj.append((c,b))`
      $\tau = 0$
    else
      $\tau += 1$
    `h.append(c)`

    $s \leftarrow$ `[c,rt,h,`τ`]`
  return `traj`

To model the RL problem, we will define the state space, action space, and the training methodology.

State: The state s_t at time t captures the spatial and temporal information of the target. The spatial information include the position r_t of the target in the

current camera FOV and the temporal information include camera history h_t, and time elapsed τ. The state vector is following,

$$s_t = (x_t, h_t, \tau). \tag{3.1}$$

The individual elements of the state space are following:

1. x_t: it is the last seen location of the target. It consists of (c, r), where c is the last seen camera and r is the spatial location of the target in camera c. To compute r, the input image is divided into a 8×8 grid, and all the cells are numbered in row-major order. The cell numbers corresponding to the target's bounding box are identified and one of these is used as r as shown in Figure 3.2.

2. h_t: it represents the history of the cameras polled by the learned policy in past N time steps, where N is the number of cameras.

3. τ: it captures the time elapsed since the target was last seen in a camera. The elapsed time is discretized with a step size of 0.25 sec and with every time step that the target is not found, the value of τ increments by 1. It resets to 0 once the target is found.

Actions: The action a_t at time t is encoded by $N + 1$ dimension vector, where N is the number of cameras in the camera network. The action $N + 1$ is selected when the policy selects no camera, i.e., the target is not visible in all the cameras.

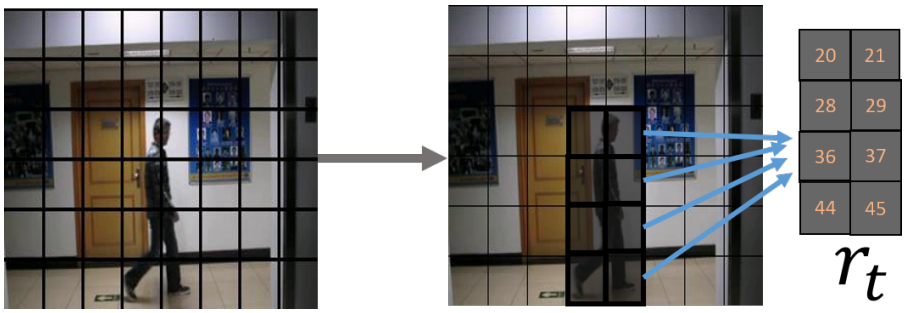


Figure 3.2: The cells in the grid capture the spatial location of the target. The image is first discretized in 8×8 grid, and all cells are numbered sequentially in row-major order. The cell number on the target position is used as the spatial location of the target named r_t at time t. One cell value among all r_t is used in the state vector.

State evolution: After deciding an action a_t, the next state s_{t+1} is decided by following state evolution function:

$$s_{t+1} = f(s_t, a_t) \tag{3.2}$$

The function appends the selected camera c_t to the camera history. If the target is found in polled camera then last seen location is updated to new (c, r) otherwise telapse is incremented accordingly.

Reward: The reward function $R(s)$ is defined for each state irrespective of the action a. At time t, it is the following:

$$R(s_t) = \begin{cases} +1 & \text{if the target is present in } s_t \\ -1 & otherwise \end{cases} \tag{3.3}$$

Training procedure: We define state-action value function Q to estimate the values (reward) of actions at a given state. The estimates will then be used to make the action selection decision. The function $Q(s, a)$ estimates the value of state action pair (s, a). The goal is to learn optimal state-action function Q^* [57].

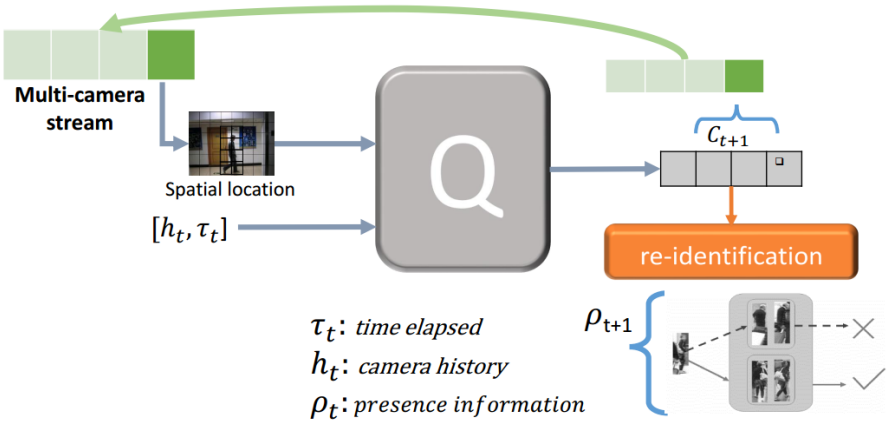


Figure 3.3: The proposed architecture using reinforcement learning. The architecture shows two blocks, block Q and block *presence*. Block Q learns a policy to select a new camera using current state and block *presence* verifies whether the target is present in the camera frame chosen.

Traditionally, the Q-functions are iteratively learned using Q-learning [57] as shown below:

$$Q(s_t, a_t) \Leftarrow Q(s_t, a_t) + \alpha\Big(R(s_{t+1}) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big) \tag{3.4}$$

Where α is the learning rate, and γ is the discount factor. We use epsilon-greedy exploration strategy [57] with epsilon annealing to explore the state-space.

Policy: The policy π selects an optimal action from the learned Q-functions. After learning, given the target state, it selects an optimal action in-state s_t as:

$$\pi_t^*(s_t) = arg\max_a Q^*(s_t, a) \tag{3.5}$$

3.2.2 System Architecture

The system architecture is shown in Figure 3.3. The architecture consists of two blocks, first, block Q which learns a policy π to select the next camera where the target will appear given target's current state. Second, the *presence* block which will verify whether the target is present in the camera selected by the policy at that time frame. The *presence* block takes as input the selected camera frame and will return 1 if the target is present along with the bounding box otherwise it returns a 0. The presence block can be implemented using person Re-ID [65]. In this chapter, we simulate the presence block with errors to falsely identify the presence. Details of this block are given in the results sections. The policy (block Q) takes as input the current state of the target (the spatial information and the temporal history) and selects a camera where the target is likely to appear. The policy is learned using Q-learning [57]. Algorithm 1 show the pseudo-code for tracking a target using the proposed method.

3.3 Experiments and Results

In this section, we present details of the dataset used, the evaluation metric and the experimental results.

Table 3.1: Details of NLPR-MCT dataset [1], which has four subsets. The table shows number of cameras (#Cameras), duration of the capture, frame rate (FPS) and the number of people (#People) captured in each subset.

	Set1	Set2	Set3	Set4
#Cameras	3	3	4	5
Duration	20 min	20 min	3.5 min	24 min
FPS	20	20	25	25
#People	235	255	14	49

3.3.1 Dataset and Evaluation Metric

Dataset: We have used NLPR-MCT data set [2] for training and testing of our proposed approach. The dataset consists of four sub-datasets each having $3-5$ cameras with a resolution of 320×240. Details of the dataset are given in Table 3.1. The dataset comprises cameras installed in both indoor and outdoor environments with significant illumination variation across different cameras. The set-1 and set-2 of the dataset have the same environment and network topology. The set-3 was captured in an office building, and the set-4 was captured in a parking area. We learn a separate policy for set-1, set-3, and set-4. Since the camera network in set-2 is same as set-1, we use the same policy for both subsets. The training and the testing sets are constructed by randomly selecting half the people for training and the remaining half for testing for each dataset. We expect the policy to implicitly learn the network topology, and so long as the network is static, the policy should work for all new, unseen target individuals. Typically, CCTV network topologies in the real-world are seldom modified.

We define evaluation metrics over the entire sequence of frames generated by the camera network. The sequence is indexed by time-steps corresponding to the time of frame capture for the cameras. Since the cameras operate

on the same frame rate for a given subset, we can ignore any synchronization errors without any significant impact on the camera selection and tracking performance.

Table 3.2: Table is showing confusion matrix of the camera selections made by the proposed policy for DB-3. Rows are the ground truth cameras and columns are the cameras polled by the policy. Values are percentages rounded off to third decimal.

$\downarrow GT/p \rightarrow$	C_1	C_2	C_3	C_4	$C_\times$
C_1	0.91	0.003	0.005	0.002	0.081
C_2	0.006	0.796	0.008	0.011	0.178
C_3	0.015	0.02	0.828	0.015	0.120
C_4	0.003	0.005	0.004	0.767	0.219
$C_\times$	0.08	0.08	0.079	0.072	0.685

Evaluation Metric: To evaluate the camera selection performance, we report camera selection accuracy, precision, and recall computed over the entire sequence of each subset. In order to consider instances when the target is not visible in any of the cameras, we introduce a dummy *null* camera and denote it by $C_\times$. Given a target, let the ground truth sequence of cameras in which it appears be contained in the vector g and sequence of cameras polled by the policy be in vector p with the i^{th} element indicated using a subscript. The Accuracy (A), precision (P), and recall (R) are defined as following for a single target

Table 3.3: Table showing average time taken (in the number of frames) by all targets from camera C_i (row) to camera C_j (column). The values (g, p) are ground truth (g) time and time taken by the policy (p) to find the target in the next camera. The values are averaged over all targets in the test set of sub-dataset 3.

Camera	C_1	C_2	C_3	C_4
C_1	(57,71)	(100,105)	(0,0)	(0,0)
C_2	(77,88)	(20,0)	(282,288)	(0,0)
C_3	(0,0)	(78,119)	(5,40)	(275,280)
C_4	(0,0)	(0,222)	(51,98)	(190,329)

$$A = \frac{\sum_i (p_i == g_i)}{\texttt{Length}(g)} \tag{3.6}$$

$$P = \frac{\sum_i ((p_i == g_i) \ \wedge \ (p_i! = C_\times))}{\sum_i (p_i! = C_\times)} \tag{3.7}$$

$$R = \frac{\sum_i ((p_i == g_i) \ \wedge \ (g_i! = C_\times))}{\sum_i (g_i! = C_\times)} \tag{3.8}$$

The final value for each of these metrics is reported as an average computed over all targets. Along with A, P, R, we also report number of frames polled (F) during an inter-camera transition of the target. It is defined as

$$F = \sum_i ((g_i == C_\times) \ \wedge \ (p_i! = C_\times)) \quad + \tag{3.9}$$
$$\sum_i ((p_i! = g_i) \ \wedge \ (g_i! = C_\times) \ \wedge \ (p_i! = C_\times))$$

F is an important measure because with a large number of frames polled, the chance of false alarms during a Re-ID query as well as the computational complexity is substantially increased. Lower the value of F is better in performance comparison.

We also evaluate the overall performance of target tracking in a camera network when our camera selection policy is used for ICT. We use the standard Multi-Camera Tracking Accuracy (MCTA) which gives a single scalar value for all components involved in multi-camera tracking, i.e., F1-score for detection, number of target handovers for single camera tracking, and the number of

Table 3.4: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for ICT alone and for both SCT with ICT.

	Set-1			Set-2		
	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$
	Inter-camera Tracking (ICT)					
Exhaustive	0.025	0.008	1.0	0.019	0.007	1.0
Neighbor	0.025	0.013	1.0	0.019	0.009	1.0
Gaussian	0.435	0.215	0.127	0.40	0.16	0.195
Proposed	0.85	0.042	0.31	0.86	0.037	0.31
	Single-camera tracking + Inter-camera Tracking					
Exhaustive	0.72	0.24	1.0	0.65	0.22	1.0
Neighbor	0.72	0.36	1.0	0.65	0.32	1.0
Proposed	0.91	0.95	0.83	0.88	0.94	0.78

handovers in inter-camera tracking. The metric is defined as

$$MCTA = \underbrace{\left(\frac{2P_T R_T}{P_T + R_T}\right)}_{F1-score} \underbrace{\left(1 - \frac{\sum_t \mu_t^s}{\sum_t tp_t^s}\right)}_{within-camera} \underbrace{\left(1 - \frac{\sum_t \mu_t^c}{\sum_t tp_t^c}\right)}_{cross-camera} \tag{3.10}$$

where P_T is the precision, R_T is recall for target IDs. The number of target-ID mismatches at time t is given by μ_t and tp_t is the number of true positives in a single camera at time t. The superscripts s and c denote the single camera tracking (SCT) or cross-camera tracking (ICT) scenario. Readers are requested to see [28, 2] for details about the MCTA metric.

3.3.2 Experiments

We design three experiments to evaluate our camera selection approach independently as well as a part of a target tracking framework. As in most tracking settings, we assume the initial location of the target to be known as given by the camera and a bounding box around the target.

Experiment-1: Evaluation of Camera Selection:

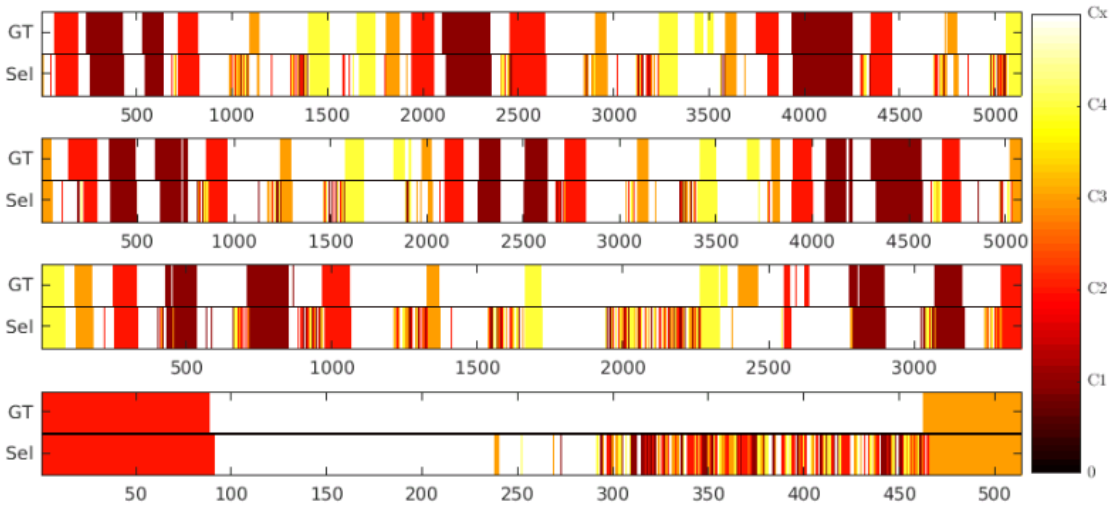


Figure 3.4: The figure shows the transitions for 4 targets in the testing set of dataset-3. GT is the sequence of cameras in ground-truth, Sel is the sequence of cameras selected/polled by the policy. Horizontal axis is the time. White color is the time when the target is transitioning between cameras and colorbar depicts the camera numbers in the plot.

Table 3.5: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for ICT alone and for both SCT with ICT.

	Set-3			Set-4		
	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$
	Inter-camera Tracking (ICT)					
Exhaustive	0.008	0.002	1.0	0.017	0.003	1.0
Neighbor	0.008	0.003	1.0	0.017	0.006	1.0
Gaussian	0.36	0.007	0.571	0.33	0.0078	0.168
Proposed	0.685	0.026	0.929	0.519	0.027	0.808
	Single-camera tracking + Inter-camera Tracking					
Exhaustive	0.42	0.10	1.0	0.56	0.11	1.0
Neighbor	0.42	0.14	1.0	0.56	0.18	1.0
Proposed	0.76	0.64	0.86	0.77	0.61	0.91

In this experiment, we initialized the state of a target with its initial location and used the learned policy to poll cameras at subsequent time steps. Based on the presence or absence of the target, the state vector is appropriately updated. In this experiment, we use the ground truth location for determining the presence of the target. We make this simplifying choice in this experiment to eliminate the uncertainty introduced due to the Re-ID performance. The policy continues polling of cameras until the target exits the camera network or

the sequence terminates. Metrics like accuracy, precision, and recall encapsulate overall performances and allow comparative analysis as shown in Table 3.4 and 3.5, which reports the camera selection performance. In addition to the proposed policy's performance, we also compare with exhaustive search and nearest neighbor search as used in multiple related works discussed in the following text. The *Exhaustive* approach is a brute-force approach which polls each camera at all time steps until the target is found in one of the cameras. The tables (3.4 and 3.5) shows that it has 100% accuracy but poor precision. The *Neighbor* approach assumes that the camera network topology is known and searches the target by polling only in the neighboring cameras. Approaches proposed in [18, 38] searches the target in the adjacent cameras and hence process the same number of frames as the *neighbor search* approach. Along with these two approaches, we also compare camera selection performance with a method proposed in [7]. The approach proposed in [7] first estimates the distribution of the camera transitions assuming the fact that the multiple targets generally follow same paths and then samples a transition time to reduce the number of frames to be processed. They estimate a Gaussian distribution and hence we named this approach as *Gaussian*. After the transition time, they start searching the target in cameras using a camera link model which will link different cameras having a path for transition. We repeated their experiment by estimating a Gaussian distribution from the train set and sampling a transition time for each person in the test set. The camera link model is used as set of neighboring cameras. The metrics computed in Table 3.4 and 3.5 are reported for two cases:

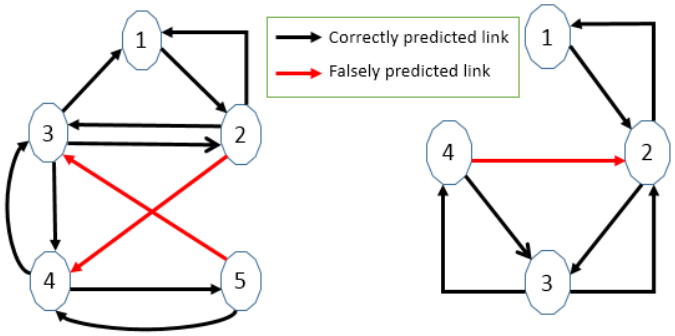


Figure 3.5: The figure shows the learned topology for set 4 (5 cameras) and 3 (4 cameras). A black arrow indicates the correct prediction and red arrow indicate a false positive.

For ICT, the metrics are computed using equations (3.6, 3.7, and 3.8) but only using the time instances when the target is transitioning from one camera to the other. For SCT + ICT, the entire sequences are used. As expected, we see that the proposed policy has better precision than the other competing approaches. The Gaussian method is excluded in case of SCT + ICT because the distribution is only defined for the ICT case.

While the A, P, and R measures indicate the overall performance of camera selection, a confusion matrix shows the pairwise miss-classification in camera selection. Based on the cameras being polled by our policy at various time steps, we report a confusion matrix for DB-3 as shown in Table 3.2.

Figure 3.4 show the sequence of cameras polled by the policy as compared to what is seen in the ground truth. Horizontal axis is time and vertical axis shows the camera schedules in ground truth (GT) and polled by policy (Sel). The dark colors are camera schedules (mapped with colormap) and white is the transition time. The figure reflects that the policy starts polling the camera early when the transition time is large and skips few frames while selecting a correct

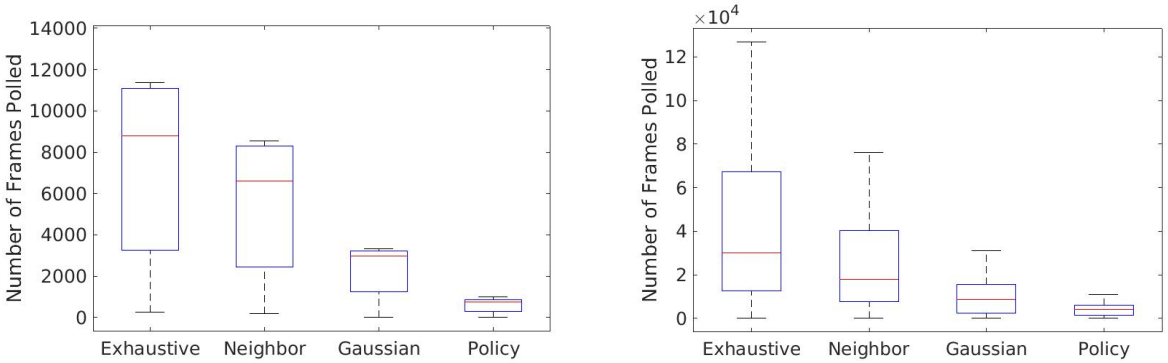


Figure 3.6: Boxplot of number of frames polled (metric F, see equation 3.9) on two datasets of NLPR dataset for our proposed policy and other baseline approaches.

Table 3.6: The table is showing average MCTA values (higher is better) for inter-camera tracking (ICT) and both SCT-ICT on the test set of NLPR-MCT dataset. The related approaches are multi-camera multi-target tracking approaches taken from the benchmark dataset [2]. The last 5 rows show the MCTA values for the proposed approach with simulated Re-ID errors from 0% to 20%.

	Inter-camera tracking (ICT)				**Single-camera tracking + ICT**			
Approach	Set-1	Set-2	Set-3	Set-4	Set-1	Set-2	Set-3	Set-4
[10]	0.9152	0.9132	0.5163	0.7152	0.8831	0.8397	0.2427	0.4357
[2]	0.7425	0.6544	0.7369	0.3945	0.7477	0.6561	0.2028	0.2650
[64]	0.6617	0.5907	0.7105	0.5703	0.6903	0.6238	0.0848	0.1830
[2]	0.3203	0.3456	0.1381	0.1562	0.8162	0.7730	0.1240	0.4637
[7]	0.9610	0.9264	0.7889	0.7578	-	-	-	-
[1]	0.835	0.703	0.742	0.385	0.8525	0.7370	0.4724	0.3778
RL+Re-ID-0	0.8210	0.7498	0.9099	0.8993	0.8235	0.7503	0.9134	0.9118
RL+Re-ID-5	0.8188	0.7481	0.8766	0.8137	0.7778	0.7064	0.7949	0.7338
RL+Re-ID-10	0.8219	0.7511	0.8848	0.7140	0.7355	0.6635	0.6791	0.6769
RL+Re-ID-15	0.8171	0.7468	0.7862	0.7128	0.7004	0.6160	0.6229	0.5879
RL+Re-ID-20	0.8203	0.7519	0.7101	0.6625	0.6281	0.5323	0.5541	0.5288

camera when target enters a new camera FOV. This is because of varying speed of different target individuals.

Experiment-2: Evaluation of Inter-Camera Tracking:

In this experiment, we evaluate only inter-camera tracking (ICT) of our proposed method to capture the performance when the target is navigating from one camera FOV to another. Unlike experiment-1, ICT includes ground truth sequence during SCT. For ICT, it is not only required to find the next camera

where the target is likely to move but also the transition time the target will take before appearing in the next camera. The transition time is critical because it will increase the number of search operations until the target is found. In this experiment, the single-camera tracking is taken from ground-truth. In our approach, we use the learned policy to select a camera at any given time using the current state of the target and when the target is located in a camera frame then ground truth results are used until it disappears from the camera FOV either due to an occlusion or a possible transition. Unlike [7], we do not model the transition time explicitly. However, the policy implicitly captures the transition time by selecting a camera at all times. The policy selects action $N + 1$ when the target is not visible in all the N cameras of the camera network otherwise it selects an action from 1 to N. The sequence of $(N + 1)^{th}$ action selection gives us the transition time of the target. Using this experiment, we will show that the learned policy can learn the camera network topology and captures the transition time of the target and hence is able to reduce the number of frames to be processed (i.e., number of Re-ID calls) at any given time. Visually, it can be seen in the Figure 3.4. The length of white color in GT is the actual transition time between any two cameras and in the same time duration the length of white color in Sel gives the length of transition time captured by the policy.

For the experiment, the state is initialized with the initial position of the target. Whenever the target moves out of the current camera FOV, the policy selects a camera (equivalently, selects an action in reinforcement learning) where the target is expected to appear. As discussed earlier, the policy selects

$(N+1)^{th}$ camera when the target is expected to be absent in all camera FOVs. The selected camera frame is then used by the *presence* block to find whether the target is present in the frame (in camera $N+1$, the target is always absent). As stated earlier, the *presence* block can be implemented using Re-ID and we will simulate different kinds of errors in Re-ID. The tracking performance is reported in terms of MCTA metric in Table 3.6. The related methods are taken from the dataset benchmark available at [2]. These approaches are for multi-camera multi-target tracking and hence we have used our policy for multiple targets to make it a multi-target tracking approach. For ICT, the tracking performance is reflected only for the duration of transition till the right camera is predicted/selected. We have simulated errors in a typical Re-ID pipeline from 0% (no error) to 20% (high error). In the table, the set-1 and set-2 show approx. same values for all percentage of errors in Re-ID and this is because the many of the target individuals are only in the single camera view and making the average value for all target high for all cases.

During training, we did not assume the camera network topology, and the policy learns the links. The transition time is recorded from the frame when the target disappears from current camera FOV to the camera frame where it is found present by the *presence* block. Table 3.3 show the time taken in the ground truth and identified by the policy for different camera transitions in dataset-3. The values in the table are the number of frames during camera transitions as found in ground truth and by the policy. Based on these transitions captured by the policy , we have made a graph of the camera network which depicts the

camera network topology. The predicted camera network topology is shown in Figure 3.5 for sub-dataset 4 (5 cameras) and 3 (4 cameras). A black arrow in the figure shows the right prediction by the policy; red arrow show a link is predicted by the policy but it does not exist in ground truth. The figure shows that the policy learns most of the links and hence the target's current state helps in finding the next camera. There are false positives and false negatives due to scheduling queries of different target individuals. Camera schedules of target 2 in Figure 3.4 shows that the transition at time frame 3500 is missed and hence this has generated a false positive in DB-3 topology.

One crucial aspect of target tracking is the number of frames polled (the number of Re-ID calls) because a large number of frames will incur a more considerable delay in locating the target. For single camera tracking (SCT), the target will be searched only in the same camera whereas, for ICT, it will be searched across multiple cameras which will increase the number of frames to be polled to find the right camera where the target has appeared. The number of Re-ID calls or the number of frames polled (F) are shown in Figure 3.6. The proposed policy queries approx. 10 times less number of camera frames compared to other baseline methods (explained earlier in experiment-1).

Experiment-3: Evaluation of Complete Pipeline: In this experiment, we evaluate our proposed policy during ICT along with single-camera tracking (SCT) for multi-camera multi-target tracking. The experimental setup is same as experiment-2 but camera sequence during SCT is taken from policy as compared to the ground truth. Table 3.6 show the comparison of our approach with

other related methods for this experiment. Unlike previous experiment, both SCT and ICT performance is compared. The Re-ID is simulated with error from 0% to 20%. The policy outperforms many methods and for set-3 and set-4, it has best best performance even at high error in Re-ID.

3.4 Discussion

This chapter presented a novel method to identify the camera schedule of a target's motion in a network of cameras. For this, we have learned a reinforcement learning based policy to select the next camera where the target is likely to appear at next time. We showed that ICT is very challenging and existing methods has to poll a large number of frames to search the target after a transition. We showed that camera selections provide better tracking performance and queries fewer frames in the camera network. We showed through various experiments that the proposed approach is also useful to capture the transition time required by a target to move from one camera FOV to other camera.

There are a few limitations of the proposed method. One, it doesn't scale to a larger camera network because the exact RL method goes out of memory. Second, the tracking performance is highly sensitive to the Re-ID performance. Along with these, the proposal needs evaluation on larger camera networks.

■

Chapter 4

Intelligent Querying in a Camera Network Using Deep Q-learning with n-step Bootstrapping

In this chapter, we will propose a camera selection approach using Deep Q-learning (or also referred to as Deep Q network or DQN), a neural network based implementation of the Q-learning algorithm. This is essential for making camera selections for larger camera networks. Using time-limits [11] and n-step bootstrapping [57], we will show that our method works effectively for making camera selection decisions. We modify the state vector proposed in the previous chapter to handle larger networks and use time-limits [11] to handle indefinite transition times, while still maintaining the MDP formulation and learn the policy using DQN.

In this chapter, we will first give a brief overview of tracking in a larger camera network, limitations of existing methods. We will then give proposed

system overview, problem formulation using Markov Decision Process (MDP), and training of the proposed method using Deep Q-learning in section 4.2. This is followed by evaluation methodology, results, and comparison with state-of-the-art methods in section 4.3. We will point out the limitation of the proposed method in section 4.4.

4.1 Introduction

Camera networks are becoming ubiquitous in smart cities where monitoring of urban environments has numerous applications like traffic management, law enforcement and security, and automated surveillance. In these scenarios, camera sensors are deployed in public spaces like road intersections, common areas in residential, commercial and government complexes to collect data, which is transmitted, stored and analysed by the government or local authorities. For example, surveillance cameras in residential and commercial complexes can be used to identify and track trespassers and unauthorized personnel or for forensic analysis during investigation.

For these applications, tracking targets across the network of cameras is important and most approaches for multi-camera tracking are driven by the state-of-the-art visual object detection, tracking and re-identification methods. While single-camera tracking poses challenges like appearance, lighting, viewpoint and background variations and occlusions, multi-camera tracking with non-overlapping fields-of-view (FOV) poses a different challenge of re-identification

of targets across cameras. Since camera networks often have units that are spatially distant, transition times from one FOV to another may take several seconds or minutes or even longer depending on the scale of the camera network. Depending on network size and the cameras' FPS, these networks generate a deluge of video frames which are potential query candidates for the re-identification module. For handling such volumes, scalable methods are of vital importance. One common approach is to select the potential camera feeds where the target is likely to be present. This approach can benefit both manual and automated surveillance as fewer frames need to be processed for tracking targets of interest. Along the same lines, we investigate *camera selection decisions* to identify the most likely camera frame where the target may reappear at the next time instance.

The inter-camera target handovers are typically resolved using visual re-identification (Re-ID) techniques, where the current template of the target is matched against all target candidates in *all candidate cameras*. Even for small camera networks with non-overlapping camera FOVs, this association problem becomes very challenging because of the non-deterministic and unknown time a target takes to transition between two non-overlapping FOVs. This uncertainty results in a large number of candidate frames, each with possibly many target candidates. Since most Re-ID or verification approaches work at an operating point chosen based on a fixed False Alarm Rate (FAR), the number of false alarms will depend on the number of frames processed for Re-ID. Re-ID false alarms could be very detrimental to the tracker's performance. Hence minimiz-

ing the number of frames that undergo a Re-ID query is critical to the tracking performance in camera networks, as well as reduce the computational complexity necessary to reduce the processing of frames not queried. An intelligent camera frame selection strategy could benefit both the accuracy and efficiency of a multi-camera target tracking system.

In this chapter, we highlight this important problem of camera selection in multi-camera target tracking. Ideally, none of the camera frames should be selected for a Re-ID query during a target transition period. Consequently, we propose to *learn* a camera selection policy that intelligently schedules Re-ID queries to resolve inter-camera handovers. We design our approach in a manner that the learning strategy directly leverages the video data and does not depend upon the network topology. We will show experimentally that our proposed method makes very few queries to the network as compared to the baseline and other competing methods used in the literature.

Based on the observation that target appearance in a camera is time-varying, it is natural to model the camera selection problem for scheduling Re-ID queries as a Markov Decision Process (MDP), which was investigated in the previous chapter by employing the Q-Learning method to exactly solve the MDP. However, exact methods are hard to scale for larger camera networks, which have larger state and action spaces. Therefore, in this chapter, we present it's extension and show that deep learning based approximate methods like Deep Q-Networks (DQN) [51] can be effectively used to scale up our camera selection approach to larger camera networks. In addition to the datasets used in the previ-

ous chapter like NLPR-MCT [2], we also evaluate the approximate approaches with larger camera networks like the Duke MTMC dataset [28]. The learned camera selection policy is used for inter-camera tracking (ICT) to generate an action that corresponds to waiting for the next time step by selecting a *dummy camera* or selecting one of the real cameras to make a Re-ID query. Finally, the policy is learned directly from the videos captured from the camera units and does not assume the knowledge of the underlying network topology. Nonetheless, in our experiments, we observe that the policy implicitly learns the network topology anyway.

The specific contributions in this chapter are:

1. We highlight the importance of camera selection decisions to enable accurate and efficient target tracking in a network of cameras.
2. We extend our approach of reinforcement learning (RL) based intelligent Re-ID query scheduling in camera networks (using exact Q-learing from the previous chapter) to use deep neural network based approximate techniques, which enables the learned policy to scale to larger camera networks where the exact methods fail.
3. We modify the state vector proposed in our previous approach in Chapter 3 to handle larger networks and use time-limits [11] to handle indefinite transition times, while still maintaining the MDP formulation and learn the policy using DQN, an approximate method (Details in Sec. 4.2.2).
4. We demonstrate over multiple real-world datasets pertaining to both indoor

and outdoor environments that the learned camera selection policy queries a very small number of frames with a small trade-off on the recall values.

4.2 Proposed Methodology

In this section, we will provide the details of the system architecture and the reinforcement learning formulation for the camera selection decision problem.

4.2.1 System Overview

Figure 3.3 on page 27 in Chapter 3 shows our system architecture, which consists of two blocks: First, block Q which learns a policy π to select the next camera where the target is likely to appear given target's current state. The second block verifies the presence of the target in the selected camera frame. In surveillance, this is usually done manually using human input or automatically using re-identification [66, 65] based approaches. We named this *presence* block. The *presence* block verifies the presence of the target in the selected camera frame and will return 1 if the target is present in the camera frame along with the corresponding bounding box location, otherwise it returns a 0. As our focus is on learning the policy for camera selection, we begin with the assumption that the presence block is perfect, and then investigate the impact of error in presence prediction. We achieve this by using ground truth labels for simulating a perfect Re-ID approach, and then induce random matching errors at different levels, in effect simulating outputs from Re-ID models at different levels of accuracy.

This setting is followed in order to systematically evaluate the strength of our camera selection policy in the presence of Re-ID noise.

The block Q, takes as input the current state (detailed in next subsection) and selects a camera index which will be polled to search the target using the presence block. The policy selects one of the $N + 1$ actions, where N is the number of cameras. The first N actions correspond to each camera and the $N + 1^{th}$ action is to be selected when the target is transitioning from one camera's FOV to another. The sequence of selected cameras gives the target's trajectory in terms of the cameras in which the target appears temporally. This is a non-trivial task due to the unknown and non-deterministic transition time of each target during camera transitions and any error in re-identifying the target will propagate to the following frames. Hence it also requires to correct any wrong re-identification made at previous timestamps. The policy is implemented using a neural network model. The network parameters are learned using deep Q-learning [51] with n-steps bootstrapping (refer Chapter 7 in the book [57]). In the subsequent subsection, we will provide details of the training and testing algorithm for camera selection decisions.

4.2.2 Markov Decision Process and Q-learning

The goal of reinforcement learning is to learn a policy that decides sequential actions specific to the target's state by maximizing a cumulative reward function [57]. Our system architecture uses deep Q-learning to learn a policy to make camera selection decisions. A decision problem can be formulated

using a Markov Decision Process (MDP). The MDP is defined by the tuple (S, A, f, R, γ), where S the set of states, A is the set of actions, $f(s_t, s_{t+1})$ is the state transition function, $R(s, a)$ is the reward function that determines the reward that the environment provides when an agent takes an action $a \in A$ in state $s \in S$ and γ is the discount factor. We define a state-action value function Q to estimate the expected value of the return for taking action a in state s by $Q(s, a)$. Return R_t at time t is defined as the discounted sum of future rewards

$$R_t = r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4.1}$$

where γ is the discount factor which is typically included to make the return bounded. We use state-action value function $Q^\pi(s, a)$ to learn the expected return starting at state s and taking action a and using policy π for the following time-steps (we will use $Q(s, a)$ in place of $Q^\pi(s, a)$ for all following text). The value function will tell us the expectation of how good (in terms of reward) the current state and action will result in future given the current policy.

$$Q(s, a) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a\Big] \tag{4.2}$$

The goal is to learn the optimal state-action value function Q^* which we learn using Q-learning because it is an off-policy and model-free algorithm.

We formulate the camera selections as a decision problem where querying each camera is considered as a separate action. As noted by related works [5], searching the target in all cameras at all times is NP-hard and therefore, this

intelligent querying becomes important to reduce the number of search operations while tracking a target across the multiple cameras. Also, the cameras in a typical camera network are deployed far apart and hence exhaustive searching may be prohibitively computationally expensive and inconsequential when the target is transitioning between cameras. To ensure this, we propose a policy that learns to select a camera where the target is likely to be present and select an action (named $C_{\times}$) to indicate that the target is transitioning. Therefore, the task is to learn a policy $\pi(s_t)$ at agent's state s_t which will give the probability of selecting a camera (equivalently selecting an action) given the current state i.e., $p(a_t|s_t)$, where a_t is the action (or equivalently camera in the context of camera selection decisions). We will show that this policy can be learned directly from data using the trial-and-error based approach i.e., by taking feedback from the environment.

However, this problem doesn't map to the MDP directly because of the target's partial observability like occlusion from other targets or the target not present in the selected camera. For example, if the policy selects camera c_i but the target is present in the FOV of camera c_j. The observation that the learning agent gets from the environment is partially observable due to occlusions and indeterminate movement of the target. For this, history of observations is included in the state vector which becomes intractable due to very large state space [57].

In addition to the observations, we keep the action history and time elapsed in the state vector. To read more about the partial observable problem, readers

are encouraged to read Sec 17.3 in [57]. The individual components of the state vector are defined in the following text:

State: The state s_t at time t captures the observations of the target and the history of cameras h_t selected by the policy, and time elapsed τ.

The individual elements of the state space are following:

1. x_t: An observation of the target's location is its spatial location in a particular camera frame, i.e., (c, b) where c is the camera index and b is the bounding box in the camera c. We keep last 3 observations of the target to estimate the next location (for example, using kalman filter). The last 3 observations form the vector x_t. In which c is encoded as a one-hot vector and b is encoded by normalizing the bounding box location i.e., x, y, w, h. (x, y) are the pixel coordinates of the upper left corner of the bounding box and (w, h) are corresponding width and height respectively. The bounding box values are normalized by dividing the pixel coordinates by the corresponding image dimensions.

2. h_t: The action at next time-steps depends on the current action and the previous actions selected by the policy. Hence, we have included the previously selected actions to the state vector. h_t represents the history of the cameras selected by the learned policy. The history of cameras is encoded as a sequence of one-hot vectors.

3. τ: It captures the time elapsed since the target was last seen in any camera. This captures the time ticks since the target was not observed. Motivated

from time-limits in reinforcement learning [11], we have included τ to work with indefinite transition times.

Actions: The action a_t at time t is encoded by $N + 1$ dimension vector, where N is the number of cameras in the camera network. An optimal policy should select an action a from the first N actions when the target is visible in the camera index a. The action $N + 1$ is selected when the policy selects no camera, i.e., the target is not visible in any of the camera.

State transition function: After deciding an action a_t at time t, the next state s_{t+1} is decided by the following state evolution function:

$$s_{t+1} = f(s_t, a_t) \tag{4.3}$$

The function appends the one-hot encoding of the selected camera c_t to the camera history vector. If the target is found in selected camera then last seen observation vector x_t is updated by including new (c, b) otherwise τ is incremented by 1.

Reward: The reward function $r(s, a)$ is defined for each state and action pair. In the previous chapter, we provided a binary reward function and here we use a dense reward. A dense reward helps because it gives the direction when a transition is ending. We have also observed that this dense reward helps improve performance by giving smaller reward to the most frequent action ($C_\times$).

Let policy selects a camera c at time t, the reward is defined as the following:

$$r(s_t, a) = \begin{cases} \frac{1}{T_c} & \text{if the target appears in c at time } T_c \\ 0.1 & \text{if the target is transitioning} \\ -1 & \text{otherwise} \end{cases} \tag{4.4}$$

Assumptions: We assume that all the cameras of the camera network are uniquely identifiable and the camera network topology doesn't change during testing phase (the CCTV network infrastructure doesn't frequently change in the real world too).

Policy: The policy π selects an optimal action from the learned Q-value functions. After learning, given the target state, it selects an optimal action using the learned Q-value function in-state s_t as:

$$\pi_t^*(s_t) = arg\max_a Q^*(s_t, a) \tag{4.5}$$

Q-learning: Q-learning is a temporal-difference (TD) learning algorithm which learns directly from state-space exploration without knowing a state-transition model. The Q-learning learns an optimal Q-value function by iteratively updating the values using the following bellman equation independent

of the policy being followed:

$$Q(s_t, a_t) \Leftarrow Q(s_t, a_t) + \alpha\Big(r(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big) \tag{4.6}$$

Where α is the learning rate and γ is the discount factor. At state s_t, the learning agent performs an action a_t and then the environment responds with a new state s_{t+1} and a reward value. We used epsilon-greedy exploration strategy [57] for state-space exploration with epsilon annealing. We are incorporating n-step rewards to update the value function.

Q-learning with n-step bootstrapping: The Q-learning update equation (4.6) updates the value function at next time using one step reward. In n-step reward, we update the value of a state after receiving rewards for n time steps. For example, taking $n = 3$, would change the Q-learning bellman equation 4.6 to:

$$Q(s_t, a_t) \Leftarrow Q(s_t, a_t) + \alpha\Big(r(s_{t+1}) + \gamma r(s_{t+2}) + \gamma^2 r(s_{t+3}) + \gamma^3 \max_a Q(s_{t+3}, a) - Q(s_t, a_t)\Big) \tag{4.7}$$

4.2.3 Camera Selection Decisions using Deep-Q Network

In the previous chapter, we proposed a Q-learning based method for camera selection decisions where we discretized the state because of a very large state space but using deep learning we can learn features even from the continuous and larger state space. Neural networks were found to map the states to reward values in many related works [51, 67, 68]. The parameters of the neural network can be updated using gradient descent based backpropagation algorithms [69]. For all implementation of exact RL method, we have used a server machine with 128 GB RAM, 5GB GPU (Nvidia Tesla K20m) and Matlab-16b [70] version. For implementation of neural networks, we have used a workstation with 8GB GPU (Nvidia GeForce GTX-1080), 16GB RAM and in pytorch. The exact RL method worked only for NLPR MCT datasets and goes Out-of-Memory (OM) for DukeMTMC dataset.

Neural network model: Our neural network model is shown in Figure 4.1. For the neural network, we will find the optimal weights which will help the learning agent to get maximum reward. For the reward based learning, we have used deep Q-learning [51] algorithm to update the neural network weights based on the reward received from the environment. The first three hidden layers of the network have *relu* activation and the last layer, outputs the Q-values corresponding to each individual action and has linear activation. The output is a $N + 1$ dimension vector, where N is the number of cameras in the camera network. Each output corresponds to an action a_i that reflects the Q-value $Q(s, a_i)$ for

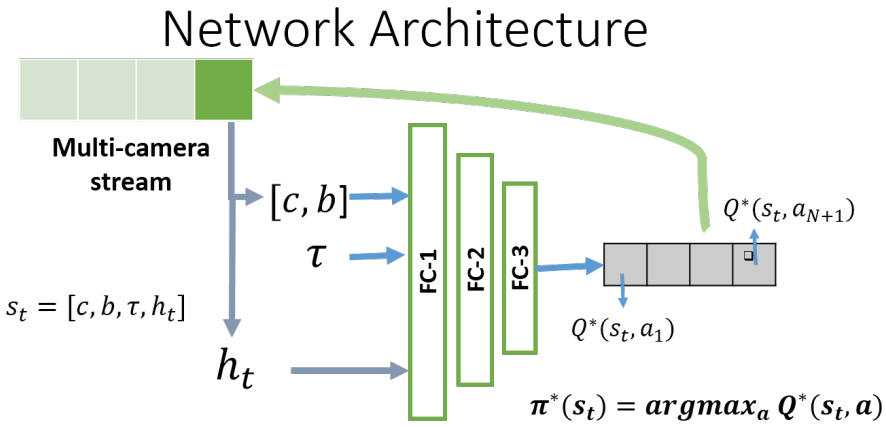


Figure 4.1: The neural network model that learns the state-action values using Q-learning. The model learns a policy that makes the camera selection decisions. This is the implementation of the Q block of the architecture shown in Figure 3.3 on page 27 in Chapter 3.

the input state s. The action corresponding to maximum Q-value of the output layer is selected by the policy (equation 4.5). The selected camera frame is then passed to the *presence* block of the system to find the bounding box location of the target in the selected camera. The state-transition function determines the next state and then the whole process repeats again.

Training procedure: During training, a replay buffer is maintained to store the transitions explored by the DQN agent. To train the network, a minibatch s sampled from the replay buffer to compute the loss. The output of the network at state s_t is $Q(s_t, a_t)\ \forall a_t \in A$ and the corresponding target is the discounted future reward for n-steps. For simplicity, taking $n = 1$, the target y_t for state s_t after receiving a reward r_{t+1} from environment is $r_{t+1} + \gamma max_a Q(s_{t+1}, a)$. We have used mean-square error to compute loss at each time-step. Hence, when action a_i is taken at state s_t, the loss (corresponding to action a_i) can be written as:

Algorithm 2 Deep Q-network training procedure with n-step bootstrapping. π is the policy to make camera selection decisions. `c,b` is the initial location of the target with c as the current camera and b as the corresponding bounding box location.

procedure TRAIN(`c,b`, π)
    Initialize replay memory M with capacity D
    Initialize deep-Q network with random weights
    $h, \tau \leftarrow$ `ZEROS` ▷ Initialize history and time-elapse with ZEROS
    $s \leftarrow$ `initialState(c,b,h,`τ`)` ▷ Concatenate location and history
    while True **do**
        `With probability` ϵ, `choose action c uniformly at random, and with probability` $1-\epsilon$, `choose action using the policy in equation` 4.5
        bb $\leftarrow$ `getBoundingBox(c)` ▷ get the bounding box from presence block
        if `rand()` < 0.5 **then** ▷ random steps are taken for exploration
            `rsteps` $\leftarrow$ `randint(20)`
        if bb is NOT EMPTY **then** ▷ update last seen observation vector if target is present in c
            $x_t \leftarrow$ `(c,bb)`
            $\tau \leftarrow$ `ZERO`
        else
            $\tau += rsteps$
        $h \leftarrow$ `updateHistory(h,c)`
        $s' \leftarrow$ `f(`x_t, h, τ`)` ▷ observe the next state and reward
        $r \leftarrow$ `getReward(s,c)`
        `Append transition` (s, c, s', r) `to replay memory M, pop last element if overflow`
        if s' is terminal **then** break
        $s \leftarrow s'$
    `Sample a random minibatch B from M`
    `For each sample` (s_i, a_i, s'_i, r_i) `in minibatch, compute the n-step target` $y_i = r_i + \gamma max_a Q(s'_i, a)$ ▷ For brevity, target value is shown for 1-step reward
    `Update the Q-network using Adam algorithm [69] on the minibatch and repeat until convergence`
    return π

$$L(s_t, a_i) = \big(Q(s_t, a_i) - (r_{t+1} + \gamma max_a Q(s_{t+1}, a))\big)^2 \tag{4.8}$$

The loss term for actions other than a_i will be zero (there are $N + 1$ actions). The term in brackets is also known as TD (Temporal Difference) error. In the loss for n-step bootstrapping, we replace the next (one) step reward with the n-step return. The step by step training procedure is shown in algorithm 2.

Algorithm 3 Camera selection decisions using deep Q learning.

procedure SELECTIONDECISIONS(c, b, π)
  $h, \tau \leftarrow$ ZEROS ▷ Initialize history and time-elapse with ZEROS
  $s \leftarrow$ initialState(c, b, h, τ) ▷ Create initial state using history and location
  while in the video sequence **do**:
    $c =$ argmax$(\pi(s))$ ▷ Choose an action using the learned policy
    Select a random c, if τ reaches the max time-limit
    b $\leftarrow$ get the bounding box using presence block
    if b is not empty **then**
      Update $x_t \leftarrow$ (c,b) and $\tau \leftarrow$ ZERO
    else
      $\tau += 1$
    $h \leftarrow$ updateHistory(h,c) ▷ Update history at every time-step
    $s \leftarrow f(x_t, h, \tau)$ ▷ Observe next states
    Append (c,b) to trajectory
  return trajectory

Note that the training procedure is same irrespective of whether the target is inside a camera field-of-view or transitioning between cameras. For training the neural network, we initialized the state vector with the initial location of the target and history vector to all zeros. The selected action (camera index) is then used to verify the presence of the target (see section 4.2.1). The state is accordingly updated using the state transition function. At any particular time, a target can see occlusion during SCT (Single Camera Tracking) and hence to simulate such cases, we randomly removed a few locations from the target's trajectory. To remove a few locations from the trajectory, we have randomly

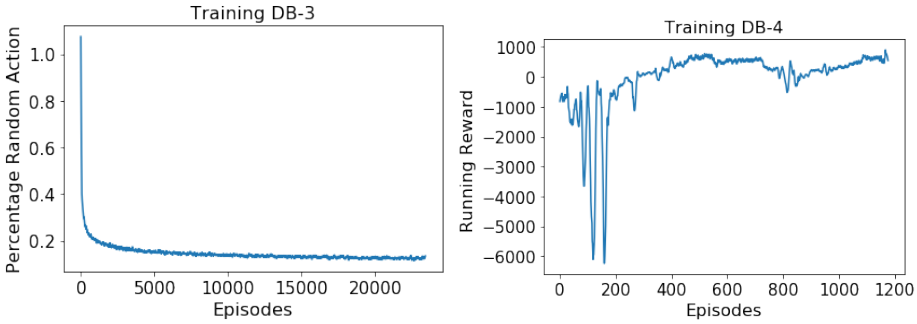


Figure 4.2: Analysis of training strategy. First, shows the varying epsilon value during training on NLPR DB-3. Second, the running reward during the training on NLPR DB-4.

sampled number of frames uniformly between $0 - 20$ (for details, see Algorithm 2). During training, τ increments by 1 when presence block cannot find the target due to removal of the location. If the target is found, τ is reset to 0. Each transition is stored in a replay memory until the end of the episode. When an episode ends, a small minibatch is sampled randomly from the replay memory for backpropagation using the Adam optimizer [69]. The training process is repeated until convergence. Instead of fixing a value for the epsilon in epsilon greedy exploration, we start with a value of 1 and decrements it as training progresses. The epsilon is set using $1/log(epoch\_number)$. The training progress is shown in Figure 4.2 which shows the percentage random actions selected using epsilon greedy exploration as training progresses. The figure also shows the reward accumulated during the training phase on NLPR Set-4. For a new target, the complete procedure to perform target tracking using the learned policy is shown in the Algorithm 3.

Table 4.1: Details of the datasets used for performance evaluation. The table shows the number of cameras (#Cameras), duration of the videos, frame rate (FPS) and the number of people (#People) captured in each dataset.

	NLPR_Set1	NLPR_Set2	NLPR_Set3	NLPR_Set4	DukeMTMC
#Cameras	3	3	4	5	8
Duration	20 min	20 min	3.5 min	24 min	1hr 25min
FPS	20	20	25	25	60
#People	235	255	14	49	2834

4.3 Evaluation and Results

In this section, we present details of the datasets used, the evaluation metric and the experimental results of the proposed architecture on the used datasets.

4.3.1 Dataset and Evaluation Metric

Dataset: We have used NLPR-MCT data set [2] and DukeMTMC [28] dataset to evaluate the proposed architecture for camera selections in multi-camera network for single target tracking. The NLPR-MCT dataset consists of four sub-datasets each having $3 - 5$ cameras with a resolution of 320×240. Details of the dataset are given in Table 4.1 and in section 3.3.1 of Chapter 3. The DukeMTMC dataset consists of 8 cameras deployed in Duke University campus. To date, DukeMTMC dataset is the benchmark dataset for multi-target multi-camera (MTMC) tracking. The details of the dataset are given in Table 4.1.

The training and the testing sets are constructed from each datasets by randomly selecting half the people for the training and the remaining half for testing. However, the evaluation benchmark of DukeMTMC dataset doesn't pro-

vide platform for camera selection performance and hence to train the policy and to evaluate the performance, we have divided the available training set into two parts by splitting person identities in two sets. Therefore, for camera selection decisions, we are reporting performance on the sub-part of the actual training set. The two sets contain mutually exclusive person identities. We expect the policy to implicitly learn the network topology, and so long as the network is static, the policy should work for all new, unseen target individuals. Typically, CCTV network topologies in the real-world are seldom modified.

We define evaluation metrics over the entire sequence of frames generated by the camera network. The sequence is indexed by time-steps corresponding to the time of frame capture for the cameras. Since the cameras operate on the same frame rate for a given subset, we can ignore any synchronization errors without any significant impact on the camera selection and tracking performance.

Evaluation Metric: To evaluate the camera selection performance, we report camera selection accuracy, precision and recall computed over the entire sequence of each subset as explained in section 3.3.1 in Chapter 3.

We perform evaluation in two parts, one for ICT alone and another for ICT along with SCT. For ICT alone case, we do not consider the frames when the target was seen in a single camera field-of-view. We also evaluate the overall performance of target tracking in a camera network when our camera selection policy is used for ICT. We use the standard Multi-Camera Tracking Accuracy

Table 4.2: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for the case of Inter-Camera Tracking (ICT).

	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$
	NLPR DB-1			NLPR DB-2			NLPR DB-3			NLPR DB-4		
Exhaustive	0.025	0.008	1.0	0.019	0.007	1.0	0.008	0.002	1.0	0.017	0.003	1.0
Neighbor	0.025	0.013	1.0	0.019	0.009	1.0	0.008	0.003	1.0	0.017	0.006	1.0
Gaussian	0.435	0.215	0.127	0.40	0.16	0.195	0.36	0.007	0.571	0.33	0.0078	0.168
Exact RL	0.85	0.042	0.31	0.86	0.037	0.31	0.685	0.026	0.929	0.519	0.027	0.808
Deep RL	0.44	0.026	0.73	0.45	0.02	0.73	0.58	0.02	0.88	0.76	0.03	0.83

(MCTA), which gives a single scalar value for all components involved in multi-camera tracking, i.e., F1-score for detection, number of target handovers for single camera tracking, and the number of handovers in inter-camera tracking. The metric is defined in section 3.3.1 in Chapter 3.

We have proposed a single target tracking approach that tracks the given target across multiple cameras whereas the related approaches on the benchmark datasets are multi-target multi-camera. To make a fair comparison with related approaches, we have created a multi-target version of our algorithm. To compute multi-target tracking results, we are running multiple pipelines of our approach for multiple targets. In our approach, the tracking performance of one target does not depend on another and hence, the approach can be easily extended to multi-target tracking problem.

4.3.2 Camera Selection Performance of the Learned Policy

In this subsection, we will describe the performance of the learned policy for camera selection decisions. There are two cases for tracking a target in a camera network. First, ICT (Inter-Camera Tracking) where the task is to identify the correct camera handovers that the target performs. Second, SCT+ICT (Single

Table 4.3: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for the case of both ICT and SCT together.

	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$
	NLPR DB-1			NLPR DB-2			NLPR DB-3			NLPR DB-4		
Exhaustive	0.72	0.24	1.0	0.65	0.22	1.0	0.42	0.10	1.0	0.56	0.11	1.0
Neighbor	0.72	0.36	1.0	0.65	0.32	1.0	0.42	0.14	1.0	0.56	0.18	1.0
Exact RL	0.91	0.95	0.83	0.88	0.94	0.78	0.76	0.64	0.86	0.77	0.61	0.91
Deep RL	0.84	0.76	0.90	0.80	0.69	0.84	0.73	0.60	0.88	0.93	0.73	0.84

Table 4.4: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for DukeMTMC dataset for both ICT alone and ICT-SCT together. The Gaussian approach is not defined for SCT+ICT case. In the table, OM signifies Out-of-Memory error.

	ICT alone			SCT + ICT		
	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$
Exhaustive	$9.6*10^{-4}$	$1.2*10^{-4}$	1.0	0.334	0.042	1.0
Neighbor	$9.6*10^{-4}$	$2.4*10^{-4}$	1.0	0.334	0.042	1.0
Gaussian	0.26	$1.9*10^{-4}$	0.58	-	-	-
Exact RL	OM	OM	OM	OM	OM	OM
Deep RL	0.81	$6.7*10^{-3}$	0.74	0.869	0.49	0.768

Camera Tracking + ICT) where the task is to identify the correct cameras when the target is moving in a single camera field-of-view along with the camera handovers.

To perform this experiment, we have initialized the initial state of the target with its initial location with history vector being all zeros. At each time-step, the learned policy selects a camera index where the target is likely to be present. The selected camera is then queried to identify whether the target is present in the selected camera field-of-view. The presence of the target is used to locate its spatial location (bounding box) in the selected camera frame. For surveillance, this task is usually performed by human agents who continuously watch the camera feed. Alternatively, this task can be achieved by re-identification based methods to automatically identify the presence of the target. Such methods use visual template matching to re-identify an object in different camera feeds given

the visual template of the target. To evaluate the camera selection decisions, we use correct presence of the target from the ground truth data. We make this simplifying choice in this experiment to eliminate the uncertainty introduced due to the re-identification performance. The policy continues polling of cameras until the target exits the camera network or the sequence terminates. The complete procedure to perform target tracking using the learned policy is shown in the Algorithm 3. For infinite horizon problems, time limits [11] in reinforcement learning have shown on various applications that randomizing the state vector (even during testing) after a time period provides better performance because larger time steps may end up in a bad state. Randomizing the state vector will help the policy to select actions from another state and eventually results in better performance. Similarly, in our case, when τ reaches a predefined maximum value, we select a random camera index to update the state vector and let the policy continue from that point to make camera selection decisions. For example, for NLPR DB-3, without using time limits, we got camera selection accuracy of 0.69 whereas by setting a time limit of 250 time-steps we got an accuracy of 0.73. We observed similar case of other datasets and used a different time limit for all datasets. All further results are reported with time limits of 800 for NLPR DB-1 and 2, 250 for NLPR DB-3, 500 for NLPR DB-4, and 600 for DukeMTMC dataset.

Metrics like accuracy, precision and recall encapsulate overall performances and allow comparative analysis as shown in Table 4.2, 4.3 and 4.4 which reports the camera selection performance on each dataset. Table 4.2 shows accuracy

(A), precision (P), and recall (R) for NLPR MCT dataset for ICT case only. Table 4.3 shows A, P, R for NLPR MCT dataset for both SCT and ICT and Table 4.4 shows the camera selection decision performance for DukeMTMC dataset for both cases, ICT alone and SCT and ICT together.

In addition to the proposed policy's performance, we are comparing the camera selection performance of the policy with three baseline approaches used in related works. The *Exhaustive* approach is a brute-force approach which polls each camera at all time steps until the target is found in one of the cameras. The table shows that it has 100% accuracy but poor precision. The *Neighbor* approach assumes that the camera network topology is known and searches the target by polling only in the neighboring cameras. Approaches proposed in [18, 38] searches the target in the adjacent cameras and hence process the same number of frames as the *neighbor search* approach. Along with these two approaches, we also compare camera selection performance with a method proposed in [7]. The approach proposed in [7] first estimates the distribution of the camera transitions assuming the fact that the multiple targets generally follow same paths and then samples a transition time to reduce the number of frames to be processed. They estimate a Gaussian distribution and hence we named this approach as *Gaussian*. After the transition time, they start searching the target in cameras using a camera link model which will link different cameras having a path for transition. We repeated their experiment by estimating a Gaussian distribution from the train set and sampling a transition time for each person in the test set. The camera link model is used as set of neighboring cameras. The met-

rics computed in each table are reported for two cases: For ICT, the metrics are computed using the same performance metric, but only using the time instances when the target is transitioning from one camera to the other. In case of SCT + ICT, the entire sequences are used. As expected, we see that the proposed policy has better precision than the other competing approaches. The Gaussian method is excluded in case of SCT + ICT, as the distribution is only defined for the ICT case. While the A, P and R measures indicate the overall performance of camera selection, a confusion matrix shows the pairwise miss-classification in camera selection. Based on the cameras being polled by our policy at various time steps, we report a confusion matrix for DukeMTMC dataset as shown in Table 4.3. The Q-learning implementation presented in the previous chapter goes out of memory for this dataset due to a very large state space. The confusion matrix is computed using deep learning based approximation of the Q-learning algorithm.

Figure 4.4 show the sequence of cameras polled by the policy as compared to what is seen in the ground truth. Horizontal axis is time and vertical axis shows the camera schedules in ground truth (GT) and polled by policy (Sel). The dark colors are camera schedules (mapped with colormap) and white color shows the length of the transition. The figure reflects the performance of deep RL policy for making camera selection decisions. One important aspect of target tracking in multiple cameras is computational time. Many related methods match target template across neighboring cameras [18, 38], all cameras [10, 5] for offline tracking. However, such approach will require a large amount of frames to

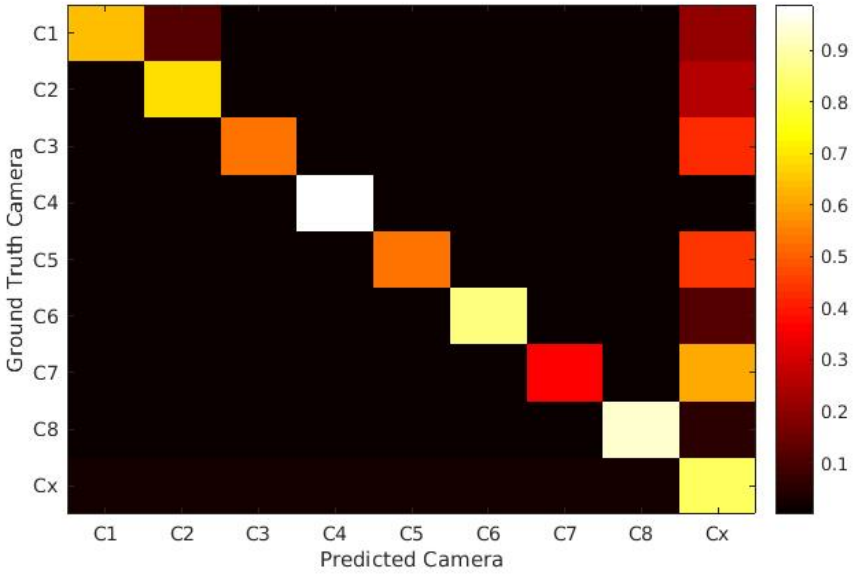


Figure 4.3: The figure shows confusion matrix of the camera selections made by the proposed policy for DukeMTMC dataset. Rows are the ground truth cameras (GT) and columns are the cameras polled by the policy. Values are percentages rounded off to third decimal.

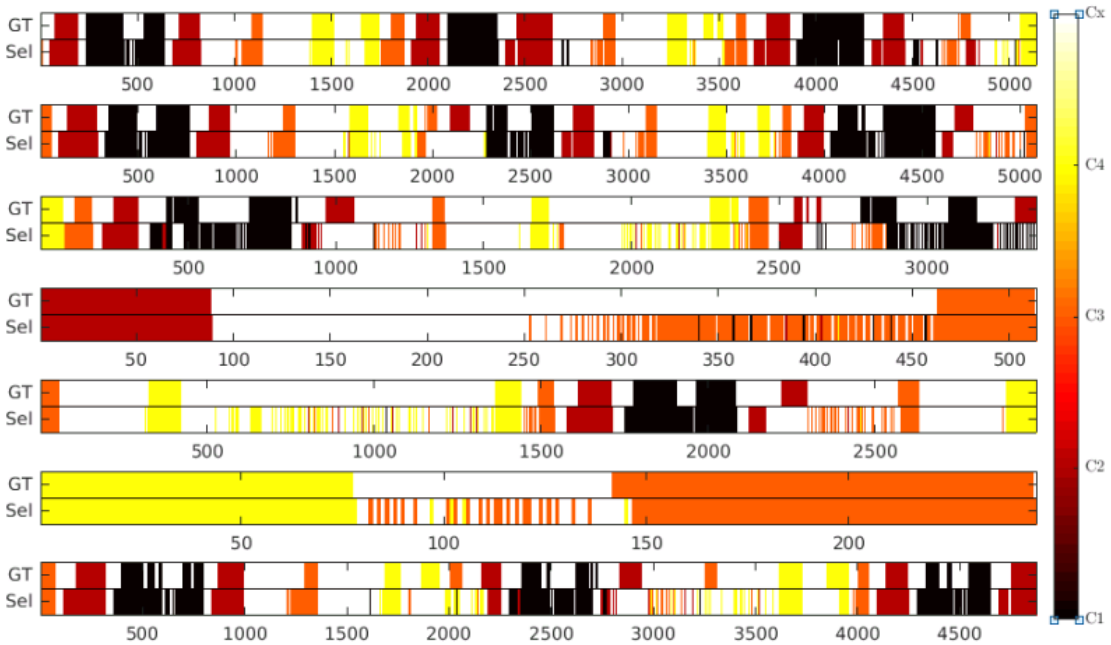


Figure 4.4: The figure shows the transitions for 7 targets in the testing set of dataset-3. On y-axis, GT is the sequence of cameras in ground-truth, Sel is the sequence of cameras polled by the policy. Horizontal axis is the time. White color is the length of the transition during camera handovers and colorbar depicts the camera numbers in the plot.

Table 4.5: The table is showing average MCTA values (higher is better) for inter-camera tracking (ICT) on the test set of NLPR-MCT dataset. The related approaches are multi-camera multi-target tracking approaches taken from the benchmark dataset [2]. The last 10 rows show the MCTA values for the proposed approach with simulated re-identification errors from 0% to 20% for both Exact RL and Deep RL implementations.

	Inter-camera tracking (ICT)			
Approach	DB-1	DB-2	DB-3	DB-4
[10]	0.9152	0.9132	0.5163	0.7152
[2]	0.7425	0.6544	0.7369	0.3945
[64]	0.6617	0.5907	0.7105	0.5703
[7]	0.9610	0.9264	0.7889	0.7578
[1]	0.835	0.703	0.742	0.385
Exact RL-0	0.8210	0.7498	0.9099	0.8993
Deep RL-0	0.9016	0.8741	0.9038	0.8074
Exact RL-5	0.8188	0.7481	0.8766	0.8137
Deep RL-5	0.7869	0.6994	0.6971	0.6118
Exact RL-10	0.8219	0.7511	0.8848	0.7140
Deep RL-10	0.7293	0.5985	0.4390	0.5673
Exact RL-15	0.8171	0.7468	0.7862	0.7128
Deep RL-15	0.7043	0.5147	0.3658	0.3946
Exact RL-20	0.8203	0.7519	0.7101	0.6625
Deep RL-20	0.6543	0.4540	0.3516	0.4680

Table 4.6: The table is showing average MCTA values (higher is better) for SCT and ICT together case on the test set of NLPR-MCT dataset. The related approaches are multi-camera multi-target tracking approaches taken from the benchmark dataset [2]. The last 10 rows show the MCTA values for the proposed approach with simulated re-identification errors from 0% to 20% for both Exact RL and deep RL implementation.

	SCT + ICT			
Approach	DB-1	DB-2	DB-3	DB-4
[10]	0.8831	0.8397	0.2427	0.4357
[2]	0.7477	0.6561	0.2028	0.2650
[64]	0.6903	0.6238	0.0848	0.1830
[1]	0.8525	0.7370	0.4724	0.3778
Exact RL-0	0.8235	0.7503	0.9134	0.9118
Deep RL-0	0.9018	0.8806	0.9058	0.7871
Exact RL-5	0.7778	0.7064	0.7949	0.7338
Deep RL-5	0.6654	0.5585	0.2210	0.4624
Exact RL-10	0.7355	0.6635	0.6791	0.6769
Deep RL-10	0.5846	0.4184	0.1333	0.3660
Exact RL-15	0.7004	0.6160	0.6229	0.5879
Deep RL-15	0.5123	0.3130	0.1176	0.3084
Exact RL-20	0.6281	0.5323	0.5541	0.5288
Deep RL-20	0.4096	0.2194	0.1196	0.2324

be processed for template matching. Using the proposed policy, this template matching will be limited to a single camera per time-step per person. In Fig-

ure 4.5, we have compared the number of frames to be processed of various such approaches. The figure shows the boxplot of F-metric scores computed over all targets using the deep RL policy and various baseline approaches on DukeMTMC dataset.

4.3.3 Impact of Camera Selection Decisions on Target Tracking in Camera Networks

Now we will show the effectiveness of the camera selection decisions to enable target tracking in a camera network. To complete the tracking pipeline, we simulate the presence block of our proposed architecture. To simulate the presence block errors in a typical re-identification pipeline are generated by wrongly identifying the target with other available objects. We will compare the performance with state-of-the-art tracking methods.

To perform this experiment, we have initialized the state vector with the initial location of the target and history vector being all zeros. The learned policy then polls a camera frame which is looked for the presence of the target using presence block (refer to section 4.2.1). Unlike previous experiment, we are simulating a real re-identification pipeline for the presence block by adding errors to the presence decision. For example, to simulate $x\%$ error in re-identification, with probability x, we are taking another target's bounding box otherwise we are using the correct bounding box of the target. Once the presence is identified, the state vector is updated using the state-transition function. The updated state vector is then used by the policy to poll another camera and the process repeats till the end of the target's trajectory or the end of the sequence. The

predicted trajectory is the sequence of (c,b) i.e., camera and bounding box values. The predicted trajectory of the target is then used to compute the MCTA metric scores. We have compared the performance of the policy with simulated re-identification errors with various state-of-the-art methods on the NLPR MCT dataset. The MCTA scores are shown in the Tables 4.5 and 4.6 for ICT alone case and SCT+ICT case respectively. In Table 4.5, we have shown MCTA values for inter-camera tracking (ICT) only where the single-camera trajectory of the target is taken from the ground-truth. In Table 4.6, shows the overall performance of the various methods i.e., during both single-camera tracking (SCT) and inter-camera tracking (ICT). The same experiments are reported by the related methods on NLPR dataset. In comparison to other methods, our approach performs better in most cases at 0% error in re-identification. For higher errors, our method (especially deep RL) starts performing worse than others. Also, the related approaches are multi-target and multi-camera (MTMC) tracking approach whereas ours is single-target and multi-camera tracking. Therefore, to make a fair comparison, we have extended our approach to MTMC as explained in section 4.3.1. Similarly, results for DukeMTMC dataset are shown in the Table 4.7.

Table 4.7: The table is showing average MCTA values (higher is better) for both SCT+ICT and ICT alone case on the DukeMTMC dataset. OM signifies Out-of-Memory error. There are no related approaches that define the tracking performance on DukeMTMC dataset using MCTA scores.

Approach	ICT alone	SCT + ICT
Exact RL	OM	OM
Deep RL-0	0.8027	0.8191
Deep RL-5	0.6438	0.6215
Deep RL-10	0.6140	0.5417
Deep RL-15	0.5879	0.4768
Deep RL-20	0.5493	0.4357

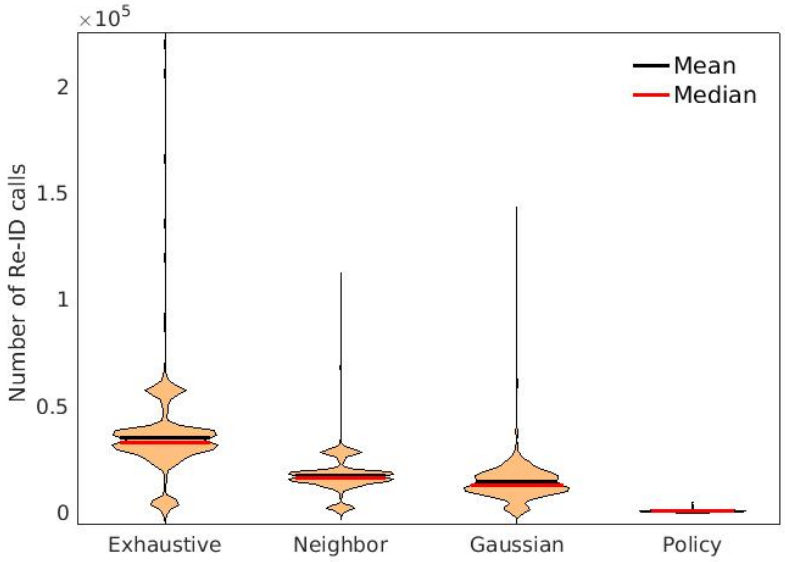


Figure 4.5: Number of frames polled (F, equation 3.9) on DukeMTMC dataset for our deep RL based policy and its comparison with other baseline approaches.

4.3.4 Comparison with State-of-the-art methods

The tracking performance and it's comparison on all dataset of NLPR-MCT dataset is shown in Table 4.5 and Table 4.6 by simulating errors in re-identification method. We are showing the comparison for tracking performance on different error thresholds of a typical re-identification method to show the impact of error in Re-ID on the camera selection performance. Similarly, for DukeMTMC dataset, we show tracking performance using MCTA metric. State-of-the-art method report ID tracking performance in terms of IDP (ID Precision), IDR (ID Recall), and IDF1 (ID F1-scores) [71]. Since original testing set of this dataset is not public making a comparison with state-of-the-art method is not fair but we are reporting average IDP, IDR, and IDF1 scores to compare the average tracking performance. The (IDP, IDR, IDF1) of the state-of-the-art methods [71, 28] on easy test set of DukeMTMC dataset is 68.3, 53.5,

60 (for [71]) and 67, 48.4, 56.2 (for [28]) whereas our method achieves 97.2, 73.7, 83.9 (with 0% error in Re-ID) and 90.8, 51.1, 65.4 (when Re-ID has 10% error). Our IDP and IDF1 is better than both the methods when we assume a 10% error in re-identification.

4.4 Limitations

The related works perform multi-camera target tracking using data association and re-identification based methods. This re-identification or association is performed either exhaustively or only in the neighboring (or linked) cameras which impacts the tracking performance. In comparison to these, in this chapter, we proposed to perform re-identification intelligently to a camera frame where the target is likely to be present. We showed through various experiments that our method presents higher tracking performance than other methods and also makes a very few re-identification queries.

The deep RL implementation make better camera selection decisions and can be used with larger camera networks. However, there are a few limitations of the proposed deep RL approach. First, the performance of deep RL approach is sensitive to errors in Re-ID. This requires investigations in training the deep learning based policy with a real re-identification so that the policy can learn how to handle errors during tracking. Second, large transition times results in a policy that has heavily imbalanced action distributions, e.g., $C_{\times}$ becoming the most frequent action. Hence, efforts should be applied in exploring methods

to handle imbalanced action space. Third, the indefinite transition time of a target makes exploration difficult in deciding whether the target goes out of the camera network or will appear again.

4.5 Discussion

In this chapter, we highlighted that Re-ID queries for target tracking across a camera network can become a performance and computational bottleneck for practical systems. In this regard, we proposed a solution that intelligently makes these queries by selecting cameras that are more likely to contain the target at a given time. We proposed a reinforcement learning based approach that learns a policy for making camera selection decisions. We empirically show on two benchmark datasets that the proposed approach achieves better camera selection performance than baseline methods. We also showed that our trained policy substantially reduces the number of Re-ID queris. Lastly, we showed that the proposed approach can be used on larger datasets like DukeMTMC dataset and showed tracking performance with varying Re-ID performance.

■

Chapter 5

Stratified Sampling Based Experience Replay

In the previous chapters, we presented a novel camera selection method and showed that it helps to efficiently track a target in a camera network. In this chapter, we will highlight one important limitation of deep Q-learning (or also referred to as deep Q network or DQN) which makes it select the most frequent action when the transitions are highly imbalanced. To handle this limitation, we will now introduce a new experience replay method (named Stratified Experience Replay, SER) to handle the limitation of deep Q-learning method. We will show that introducing this method significantly improves the camera selection performance in various camera networks. In this chapter, we will first give the motivation in section 5.1 to use a different experience replay method by highlighting the indefinite transition time degrades DQN performance. We will then talk about the literature of different RL and different experience replay methods in section 5.2. This is followed by formulation of the problem as MDP

in section 5.3 and the proposed experience replay approach in section 5.4. In section 5.5, we present difference experiments to validate our claims.

5.1 Introduction and Motivation

Reinforcement learning (RL) combined with deep learning has achieved great success in learning control policies for tasks with large state space, e.g., in game playing [72], different robotic tasks [73, 74], etc. Recently, these deep RL frameworks also showed efficacy in learning control policies for various real applications like urban traffic control [75], camera selection [9], planning [76] in both model-free and model based settings.

Many of the deep RL methods require a buffer (often called as replay memory) to store state transitions during agent-environment interaction so that at train time, a diverse set of transitions can be sampled from this replay memory to construct a better minibatch for backpropagation. Experience replay (ER) is one such approach that help these methods to store and reuse the past information for learning decision policies and plays a crucial role in stabilizing learning of approximate solutions using deep neural architectures [72]. Usually, these stored state transitions are sampled uniformly to create a minibatch, however, it has been noted that uniform sampling from the replay memory may fail to create a diverse minibatch, in turn limiting the generalizability of the underlying neural network [77]. Therefore, it becomes essential to strategically sample *important* experiences from the replay memory. To ensure sampling of important expe-

riences, many variants of ER [78, 77, 79] are proposed. A similar behavior is observed in the supervised learning domain [80] when the dataset is imbalanced. To handle the imbalanced data, a common approach is to present fewer samples of what the model has already learned well and more samples of what it has not. Inspired by the observation that frequent repetition in imbalanced dataset help effective learning, we hypothesize that the model should be exposed to such experiences more regularly. This repetition was achieved by performing dynamic sampling to handle data imbalance in a supervised learning setting in [81].

Tracking targets across a network of cameras is one such practical application, where state transitions could be heavily imbalanced. Most practical camera networks have cameras placed in unconstrained environments with varying lighting and camera orientations as well as non-overlapping fields of view (FOVs). In order to handle target handovers, as it moves from one FOV to another, visual re-identification (Re-ID) approaches are used for data association. In the previous chapters, we pointed out that a relatively unexplored, yet an important aspect of multi-camera tracking systems is the problem of selecting cameras to schedule Re-ID queries. The selection of cameras and the frequency of such queries determines the number of false alarms, which in turn directly impacts the tracking performance. We also demonstrated that using an RL framework outperforms previous model based approaches to determine which cameras to select for a Re-ID query. To appreciate the challenge in camera selection, consider the example in Figure 5.1. In a typical setting, there are multiple targets ($P_1 - P_3$) moving across the FOVs of different cameras

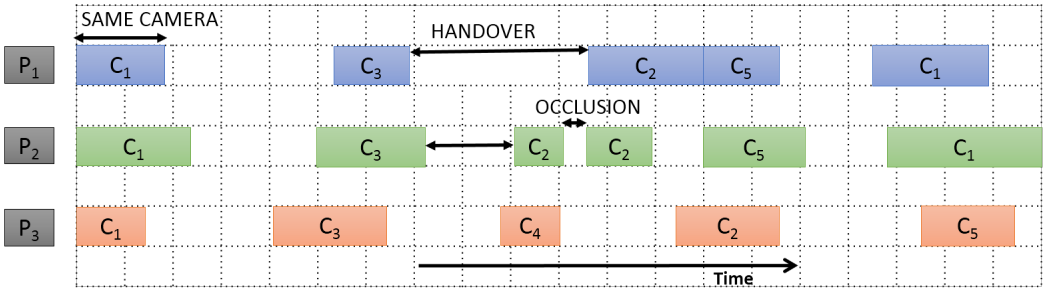


Figure 5.1: Example camera transitions for 3 targets P_1, P_2, and P_3.

($C_1 - C_5$). As a target transitions from one camera's FOV to another, a Re-ID query needs to be made to resolve target handovers. However, as shown in Figure 5.1, P_1 may take longer to transition from C_3 to C_2, while P_2 may have a shorter transition time, implying the target would not be visible for an indeterminate duration. A similar issue may arise due to occlusion of the target while it still remains in the FOV (P_2 in C_2), thus making camera selection relevant for occlusion handling. This non-deterministic nature of the transition time makes the camera selection problem a crucial one. Selecting a spurious camera (where the target is not present) increases the risk of a false alarm by Re-ID and therefore is detrimental to the overall tracking performance.

In this work, we point out that the number of camera handovers are fewer than the number of instances when the target is not visible in any camera (i.e., selecting action $C_\times$). It generates an imbalanced action space and is an important observation with respect to training a deep RL model as it requires appropriately handling of imbalanced transitions during experience replay. Our proposed approach, referred to as Stratified Experience Replay (SER) resolves this challenge by sampling in the imbalanced replay memory created by the different episodic runs of the agent-environment interaction.

In this context, our specific contributions are the following:

1. We propose a novel experience replay method to segregate transitions into multiple replay memories. Our investigations show that stratified sampling helps learning a better policy for camera selection in a camera network.
2. We will show that existing replay methods impact the minibatch and eventually effects the performance of the RL algorithm.
3. We compare the performance of DQN with our SER and other ER methods on various camera networks. We also compare performance with state-of-the-art camera selection methods on the same datasets.

5.2 Background and Related Work

Experience replay has been incorporated in many deep reinforcement learning methods to memorize and replay past experiences of the agent-environment interaction. It has been observed that ER stabilizes the training process by breaking the temporal correlations of the sequential online transitions [72]. Uniform sampling is the most common approach for sampling the transitions from the replay memory. However, researchers have demonstrated that uniform sampling fails to create a diverse minibatch for many applications [77] and hence the RL algorithms fails in such scenarios.

To tackle this problem, many variants of ER have been proposed aiming to create a diverse minibatch. Schaul et al. [77] proposed prioritized experience

replay in which each sample is assigned a probability of sampling based on the ability of approximation architecture to correctly predict the Q-estimate. A transition is sampled from the replay using the assigned probabilities. Many other approaches are known to exist which indirectly prioritize the transitions by forgetting and remembering transitions [82]. Zha et al. [78] proposed experience replay optimization (ERO) where a separate a replay policy was learned to decide which samples to select to create a minibatch. In [76], a re-weighted experience model for planning domains for incremental and data efficient learning was proposed. Foerster et al. [83] proposed a variant of ER for multi-agent environment for independent Q-learning. On the other hand, Zhao et al. [84] proposed another ER method for multi-goal environment using maximum entropy. Our empirical results show that these ER methods do not provide better sampling when the replay memory is imbalanced.

It has been shown that parallel workers for on-policy RL algorithms like PPO [85] provide better performance than using ER with off-policy algorithms [86]. In this chapter, we demonstrate that when SER applied to DQN, it performs better than PPO for our target application of camera selection. Additionally, we empirically found that SER applied to DQN learns a better policy than other ER methods. We demonstrated our results on various camera networks datasets which pertain to many real environments like parking, office building, university campus, footpath etc.

We have shown in the previous chapters that the camera selections are important for multi-camera target tracking. In Chapter 3, we used hand crafted

features for state representation and using table-based Q-learning to learn a policy which doesn't not scale to camera networks with more than 6 cameras. In Chapter 4, we proposed an extension using deep-Q learning and n-step bootstrapping. In this chapter, we will show that deep Q-learning with proposed stratified experience replay performs better than both of the methods.

5.3 Formulation as an MDP

In the previous chapter, we modeled the camera selection problem as a Markov Decision Process (MDP), which was solved using deep Q-Learning. Here, we propose changes in the reward structure and the state variable for learning a better policy.

To formulate the camera selection problem as an MDP, the initial location of the target in terms of its in-frame location and the corresponding camera is provided, which is used to create the state vector. The learned policy uses this state vector at a given time step to select an action that identifies a camera that may contain the target. The action may correspond to one of the N cameras, for which a Re-ID query can be made or it may correspond to the *null* camera $C_{\times}$ where the policy decides to skip the Re-ID query. A reward is assigned based on whether the Re-ID query was successful or not, thus penalizing incorrect camera selections where the Re-ID would fail. We emphasize that like in [9], our focus is on camera selection and not on the specific Re-ID algorithm, and hence we use ground-truth annotations to determine the failure or success of a Re-ID

query. Based on this agent-environment interaction, we formulated an MDP $(S, \mathcal{A}, f, R, \gamma)$, where S is the state space, $\mathcal{A}$ is the action space, $f(s_t, s_{t+1})$ is the state transition function, $R(s, a)$ is the reward function and γ is the discount factor. Each element is described below:

State: As the target is not always observable, e.g., during transitions or occlusions, the state at time t captures three elements to handle this partially observable state of the target:

1. x_t: the last observed location of the target and is given by (c, b), where c is the camera index and b is the in-frame bounding box within camera c. c is encoded as a one-hot vector. The vector $b = (x, y, w, h)$ is the bounding box location with (x, y) as the top left pixel coordinates and (w, h) as the corresponding width and height respectively. The values are normalized in the range $(0, 1)$ by the image size. Additionally, we also include the first order difference in the bounding box of the target, i.e., $\Delta b_t = b_{t+1} - b_t$, which captures the direction of motion and b_t is the bounding box location of the target at time t. We include these variables in to the state vector, to make the Markovian assumption on the location of the target.

2. h_t: the action history that maintains a list of previously selected actions by the policy. The history is stored as a list of cameras encoded as a one-hot vector.

3. τ: This is the time-elapsed vector that captures the time since the target's most recent observation by the agent in any camera.

Action: there are $N+1$ actions, where N is the number of cameras in the camera network. The $N+1^{th}$ action, denoted by $C_{\times}$, implies the target is transitioning and is not visible in any of the cameras.

Reward: Let y_t is the correct camera at time t and agent selects an action a_t in state s_t at time t. The reward function is defined as:

$$r(s_t, a_t) = \begin{cases} +1 & \text{a}_t = y_t \text{ and } \tau > 20 \\ +0.5 & \text{a}_t = y_t \text{ and } \tau \leq 20 \\ 0.01 & a_t = y_t = C_{\times} \\ -1 & \text{otherwise} \end{cases} \tag{5.1}$$

where y_t is the correct camera where the target is present and τ is the transition time since last appearance of the target. This reward function is adopted to counter for the stark difference in frequency of the action $C_{\times}$ versus other camera selections, which are relatively rare in real world camera networks. A similar observation is made in [87]. A rare transition is the transition when the target appears and disappears from a camera field-of-view. We point out that the number of instances of camera handovers (instances of appearing or disappearing for a camera) are fewer than the number of instances when the target is not visible in any camera (i.e., selecting action $C_{\times}$). It generates an imbalanced action space. In this work, the instances of appearance and disappearance of a target are termed as rare.

State transition function: After deciding an action a_t at time t, the next state

s_{t+1} is decided by the state transition function. In the next state vector, the selected action is appended to the history vector. If the target is absent, τ is incremented by 1, otherwise it is reset and the observation vector is updated with the new camera index c, location b and Δb.

5.4 Proposed Approach

In this section, we will describe the problem statement and the proposed experience replay method.

5.4.1 Problem Statement

We model camera selection decision as a finite horizon discounted sum reward problem, where an RL agent is responsible for deciding the presence of a target given a camera frame at a discrete time step t. At each time step, the agent receives the location of the target to be tracked contained in s_t. The agent needs to select one of the cameras represented using the action space $\mathcal{A} = \{0, 1, \ldots, N, C_{\times}\}$. The agent interacts with the environment $\mathcal{E}$ by selecting an action $a_t \in \mathcal{A}$. As a consequence, the environment $\mathcal{E}$ transitions into next state s_{t+1} and returns a 3-tuple containing next state, reward and termination status of the episode represented as (s_{t+1}, r_{t+1}, d_t). This approach of modeling the camera selection problem, leads to a finite Markov Decision Process (MDP) and permits the use of standard Reinforcement Learning algorithms.

We define the optimal Q-value of a state $Q^*(s, a)$, as the maximum expected

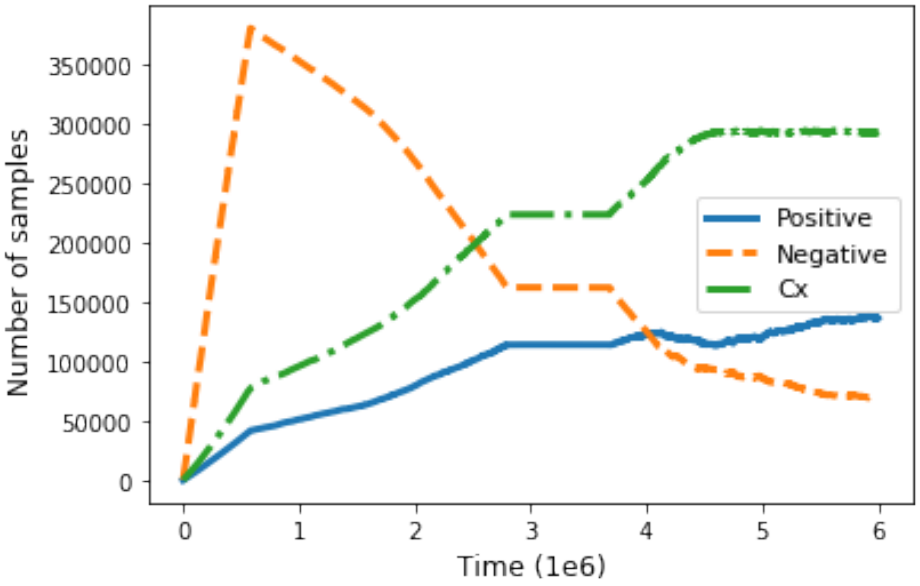


Figure 5.2: The figure shows the training time on x-axis and the number of samples of each transition type on y-axis. The transition type is shown in the legend. The replay memory is dominated by the most frequent action $C_{\times}$.

discounted sum reward which agent can collect with its interaction to $\mathcal{E}$ over the set of all possible admissible policies π, after performing action a in state s. As our state-space can be very large, we represent this function using a paramertized function $Q^*(s, a|\theta)$ such as deep neural networks.

According to *Bellman equation*, $Q^*(s, a|\theta)$ can be written in a recursive form as shown in eqn. 5.2, which is approximated by minimizing the expected loss over the collected transitions. However, the transitions are correlated in time. Therefore, a replay memory of size R is used, which stores the last $|R|$ transitions. A batch of size $\mathcal{B}$ transitions is sampled randomly from this replay to break the correlation. Deep Q learning is widely used for game playing in reinforcement learning [51].

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{\bar{a}} Q^*(s_{t+1}, \bar{a})|s_t, a_t\right] \tag{5.2}$$

5.4.2 Proposed Experience Replay Approach

Agent stores the visited transitions in a replay memory from which transitions are replayed later for learning. For learning, a diverse minibatch must be sampled. However, for camera selections, there is a large dominance of one action. Figure 5.2 shows how the replay is populated during the training process for NLPR set-3. The figure shows the training time on x-axis and the number of samples of each transition type on y-axis. The transition type is shown in the plot legend. We can see that with time, the replay gets dominated by the transitions of action $C_{\times}$. Please note that state-of-the-art replay methods use single replay memory with recommended size of 10^6. Initially during exploration, the replay gets filled with negative reward transitions but it has to learn more of $C_{\times}$ transitions and hence over time the learning is dominated by such frequent transitions.

It makes the neural network training biased toward the most frequent action, which is $C_{\times}$. Therefore, an optimal policy cannot be learned by the neural network because of the poor minibatch presentation to the network for learning. The similar behavior is observed in the supervised learning (SL) domain [80] to handle the imbalanced data for classification. In SL, the minibatch is often created by undersampling the frequent class and oversampling the rare class. General practice is to present fewer samples of what the network has learned and more samples of what the network needs to learn. It cognates to human learning where human tends to learn some tasks very easily if it gets repeated more often.

Hence to learn a new task, they need exposure to the difficult task regularly. [81] showed this repetition by performing dynamic sampling to handle data imbalance in supervised learning.

Algorithm 4 DQN with Stratified Experience Replay.

procedure SER
  Initialize experience replay memories R_+, R_-, R_f.
  Initialize estimate of state-action value function Q with random θ.
  Initialize target function Q_T with weight $\theta^- = \theta$.
  while `episodeCount` $<$ `numEpisodes` **do**
    Initialize start state $s_0 =$ *reset()*
    while *episodeSteps* $<$ *numSteps* **do**
      Select an action a_t using ϵ-greedy policy using $Q(s_t, a|\theta_t)$.
      Execute the action a_t in *environment*.
      Observe next state s_{t+1}, reward r_{t+1} and episode termination status d_t.
      Create transition $\tau = (s_t, a_t, r_{t+1}, s_{t+1}, d_t)$.
      if $r_{t+1} < 0$ **then**
        Store transition τ in R_-.
      else
        if $r_{t+1} > 0.1$ **then**
          Store transition τ in R_+.
        else
          Store transition τ in R_f.
    Sample a batch $\mathcal{B}$ from replays R_+, R_-, R_f.
    for $b = (s, a, r, \bar{s}, d)$ in B **do** $d == True$
      set $y = r$
      set $y = r + \gamma \max_{\bar{a}} Q_T(\bar{s}, \bar{a}|\theta_t^-)$

      Perform gradient descent on $\{y - Q(s, a|\theta_t)\}^2$
    Copy $\theta^- = \theta$, after every C steps.

We observed that ER methods that use single replay gets dominated by the most frequent action. As also observed by [78], the temporal difference error is not the right criterion to decide the probabilities for the sampling of the stored transitions. Therefore, we have segregated transitions into multiple replay memories to enable sampling of all kinds of transitions to create the minibatch. Also as observed in supervised learning, we need present the rare transitions more often to the network. To ensure efficient sampling of rare and other transi-

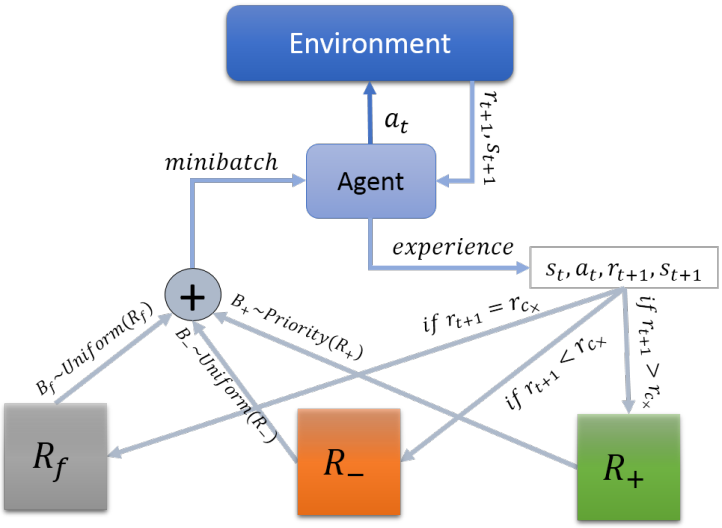


Figure 5.3: Overview of the proposed experience replay method. R_f, R_-, and R_+ are the different replay memories to store frequent, negative reward, and positive reward transitions respectively.

tions, we segregate transitions in different replay memories to compensate for searching in significantly large replay memory. We create three replay memories named R_f, R_+, and R_-. Given the three replay memories, we sample a minibatch $\mathcal{B} = (s, a, \bar{s}, r)$ which is is used to generate an empirical estimate of the expected loss $L(\theta_t^i)$, as shown in eqn. 5.3 as commonly used in deep learning [63, 52].

$$L(\theta_t) = \frac{1}{|\mathcal{B}|} \sum_{i=0}^{|\mathcal{B}|} [y_i - Q^*(s, a|\theta_t)] \tag{5.3}$$

where $y_i = r + \max_{\bar{a} \in A} \gamma Q^*(\bar{s}, \bar{a}|\theta_t)$ and $\mathcal{B} = \{\mathcal{B}_f \in R_f \cup \mathcal{B}_+ \in R_+ \cup \mathcal{B}_- \in R_-\}$.

The overview of the proposed method is shown in the Figure 5.3. We create 3 replay memories for storing the transitions from agent-environment interaction. The multiple replays are; R_f, which stores the transitions which pertain

frequently occurring action $C_{\times}$; R_{-}, which stores the transitions that receive a negative reward, and R_{+} which stores the transitions that receive a positive reward. In the system overview, the agent selects an action a_t in the environment and then the environment returns the corresponding reward r_{t+1} and the next state s_{t+1}. This transition $(s_t, a_t, s_{t+1}, r_{t+1})$ is stored in a replay which satisfies the above criteria for segregation. For learning, a minibatch is prepared from the stored transitions in the multiple replays. The transitions are sampled uniformly from R_f and R_{-} replay. R_{+} stores both rare and other positive reward transitions, it follows a prioritized sampling. For priority sampling, a higher weight is assigned to the rare transitions and a lower weight to other transitions. A probability value is assigned to i^{th} transition as $\frac{w_i}{\sum w_i}$, where w_i is the weight assigned to the i^{th} transition. The minibatch is used to learn the policy. The training pseudo-code is shown in Algorithm 4 which takes the training target trajectories and learns a policy to select cameras for each time instant. The training algorithm is a deep Q-learning method [88] with a target network to avoid maximization bias.

5.5 Experiments and Results

In this section, we will be comparing performance of different ER methods applied to off-policy DQN. We will also show that state-of-the-art replay methods fail to create a good minibatch which eventually impacts the policy.

5.5.1 Experimental Setup

We evaluate our approach on five camera network datasets collected in an indoor office building, outdoor parking space, footpath, and duke university campus. These datasets are available in [2, 5] We compared our approach against the following methods:

- PPO [85]: On-policy RL algorithm.
- ER-unif [72]: DQN with Experience replay, where data is sampled uniformly.
- PER [77]: Prioritized experience replay which prioritizes the transitions in the replay memory using temporal difference error.
- ERO [78]: Learning a policy to sample batch from replay buffer.
- CamSel [9] and nSteps [63]: state-of-the-art camera selection method using reinforcement learning for querying in camera networks. In nSteps, we investigated DQN based methods with n-step bootstrapping to learn the long transition times.

Implementation: We have implemented the DQN algorithm in Pytorch on a server with 128-GBs of RAM and 11-GB Nvidia RTX 2080 Ti GPU for training. For PPO implementation, we utilized the `rllib` library build upon ray framework [89] with TensorFlow as the backend to train the policies. To accelerate the learning process, we utilized 16-cores of the machine, each running three instances of an environment. For the proposed method, we used three replays

to separate the rare transitions, where the size of each replay was set to 10^3 acquired via hyperparameter tuning. As recommended, replay memory size for other ER methods is set to 10^6. For DQN, we used the epsilon-greedy policy as an exploration strategy during training. To train the model, we employ a deep neural network with three hidden layers (4096-1024-256) to approximate the DQN based policy. For other replays, we used a grid-based hyperparameter tuning to search the hyperparameter space. The subsequent sections present the performance on the best parameters.

Performance Metric: We use accuracy (A), precision (P), and recall (R) metric to quantify the performance for selecting cameras. $C_{\times}$ represents the instances when the target is not visible in any of the cameras. Let for a specific target, vector g hold the true sequence of cameras in which the target appears,

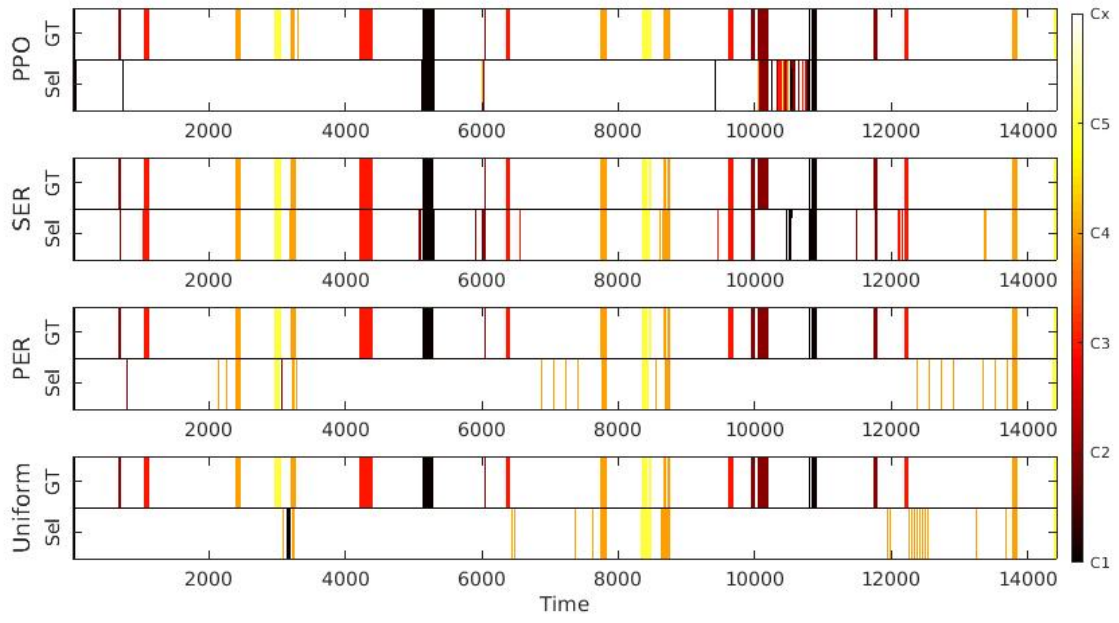


Figure 5.4: Qualitative performance of different ER methods. In the figure, each color represents a specific camera FOV and time gap between colors show the transition time of the camera handover.

Table 5.1: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the different methods on various camera network datasets. SER is our proposed experience replay method.

	Set-1			Set-2			Set-3		
	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$	$A\uparrow$	$P\uparrow$	$R\uparrow$
PPO	**0.95**	0.94	0.93	0.94	0.94	0.91	0.80	0.73	0.82
CamSel	0.91	**0.95**	0.83	0.88	0.94	0.78	0.76	0.64	0.86
nSteps	0.84	0.76	0.90	0.80	0.69	0.84	0.73	0.60	0.88
ER (Unif.)	**0.95**	0.94	0.95	0.94	**0.95**	0.93	0.82	0.73	0.75
PER	0.92	0.91	0.88	0.91	0.90	0.86	0.87	0.81	0.89
ERO	0.90	**0.95**	0.89	0.92	**0.95**	0.90	0.86	**0.83**	0.83
SER (ours)	**0.95**	0.92	**0.97**	**0.95**	0.93	**0.96**	**0.89**	0.81	**0.93**

and p contains the cameras selected by the learned policy. The performance metrics are defined in section 3.3.1 in Chapter 3.

For inter-camera tracking (ICT), we define the measure Percentage Camera Handover (PCH) as the percentage of target transitions (from Camera C_i to C_j, $i \neq j$) that are correctly detected by using the learned policy. Missing more target transitions hurts overall tracking performance, and increases the chance of not finding the target again. This is a crucial metric to compare the number of transitions recovered. Higher value is better for this metric in performance

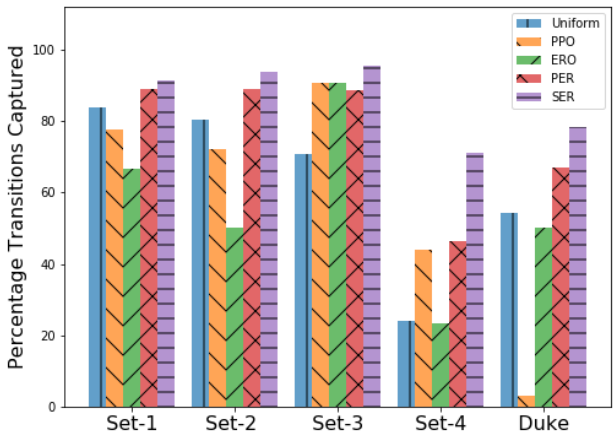


Figure 5.5: Percentage Camera handovers (PCH) (higher is better) captured by different ER methods. SER is largest for all datasets.

Table 5.2: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the different methods on various camera network datasets. SER is our proposed experience replay method.

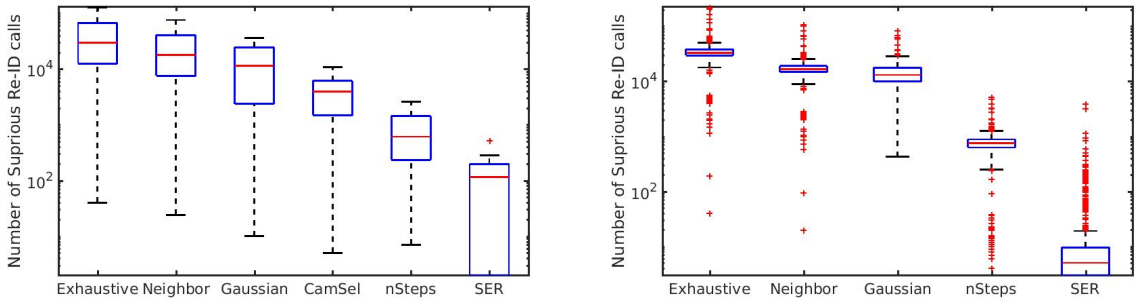
	Set-4			Duke MTMC		
	$A \uparrow$	$P \uparrow$	$R \uparrow$	$A \uparrow$	$P \uparrow$	$R \uparrow$
PPO	0.90	0.70	0.72	0.92	0.81	0.49
CamSel	0.77	0.61	**0.91**	Out of Memory		
nSteps	0.93	0.73	0.84	0.869	0.49	0.768
ER (Unif.)	0.90	0.82	0.63	**0.96**	0.89	0.73
PER	0.87	**0.84**	0.65	0.95	0.85	0.81
ERO	0.89	0.81	0.60	0.94	**0.94**	0.53
SER (ours)	**0.92**	0.74	0.84	**0.96**	0.84	**0.87**

comparison.

5.5.2 Performance comparison

Table 5.1 and 5.2 shows the camera selection performance of our proposed method and other RL based approaches. There are five datasets for comparison, Set-3 consists of four cameras (hence five actions) in an indoor office building where target transition times are not very long. Consequently, we see approaches other than SER too were able to learn many transitions for this dataset. The camera selection method [9] fails to learn a policy for the Duke dataset (8 camera network), because it stores the Q-values in table and goes out of memory for larger datasets. SER significantly outperforms all the ER methods and the on-policy RL algorithm PPO in terms of A and R on all datasets, with the exception of Set-4 where we are second. The proposed approach also outperforms our previous work where we use a DQN architecture with n-step bootstrapping.

Figure 5.4 shows the qualitative results of our proposed method and its comparison with other ER methods on Set-4. The figure shows the camera transi-



(a) (b)

Figure 5.6: Number of spurious frames polled (F in equation 3.9) on a NLPR_Set4 dataset and b DukeMTMC dataset. The figure also shows the comparison of our proposed policy and other baseline approaches

tions of a particular target for various methods. For each method, there are two sequence of cameras (ground truth is shown by the sequence GT and cameras selected by the policy are shown by Sel). In a sequence, different cameras are shown in different colors with one color for each camera (refer the colorbar for camera index). The camera sequence shows that the length of transition is very large when compared to the total length of in-camera instances. SER enables the RL agent to capture most of the transitions whereas uniform sampling could not capture many transitions and hence it fails heavily in this case. PER captures many transitions and PPO fails to capture many transitions. PPO learns a stochastic policy and hence it's behavior is more random whereas for SER the policy start early prediction of the right camera. To quantify the average number of transitions learned by different methods, we plot the bar graph of the average number of transitions captured by each method in Figure 5.5. This figure shows that SER outperforms all other methods and captures the maximum number of transitions. The other approaches fail because the replay is highly imbalanced. PER performs better on first three datasets but fails on Set-4 and Duke dataset because of very large transition time of the targets. The Set-4 shows the lowest

performance due to the highly dynamic parking area environment.

Figure 5.6 shows the number of re-identification queries made by different methods on the NLPR-Set4 and DukeMTMC datasets. The figure shows the boxplot of the number of re-identification queries (metric F in equation 3.9 on page 31 in Chapter 3) by each target in the corresponding dataset. We show a comparison with the baseline methods *Exhaustive*, *Neighbor*, and *Gaussian* which are the the different ways of querying the camera network as specified in [9]. We also compare the querying performance with our previous methods [9, 63] and *SER* is the proposed method with SER. The figure shows that our learned policy performs better than all baseline and related methods.

5.5.3 Analysis of Sampled Transitions

For further understanding of the proposed method and earlier results, we store the diversity of minibatch at each training epoch. To measure diversity, we compute the number of transitions corresponding to each reward. Again we make three categories, positive reward transitions, negative reward transitions, and $C_\times$ transitions. The diversity of minibatch of each replay method is shown in Figure 5.7 where for each method the diversity in terms of the three categories is shown as the training progresses. The results are shown for Set-3. Uniform sampling makes the learning highly biased towards $C_\times$ whereas PER is somewhat stable for this set and hence captures a good number of transitions (refer Figure 5.4). However, it also fails to create a diverse minibatch on Set-4 (Figure 5.7). This justifies that having a poor minibatch effects the policy learning

for the off-policy RL algorithm. Please note that for SER minibatch stays static, that means the number of transitions of each reward stay same throughout the training process and hence diversity plot is not shown. This stable minibatch helps in learning larger number of rare transitions.

We also learn a stochastic policy using PPO algorithm for our problem. However, we found that the sampling from a stochastic policy leads to unnecessary switching of actions. To validate our claim, we calculated the mean entropy of the learned policy and found it to be much higher than true entropy, computed using the ground truth values.

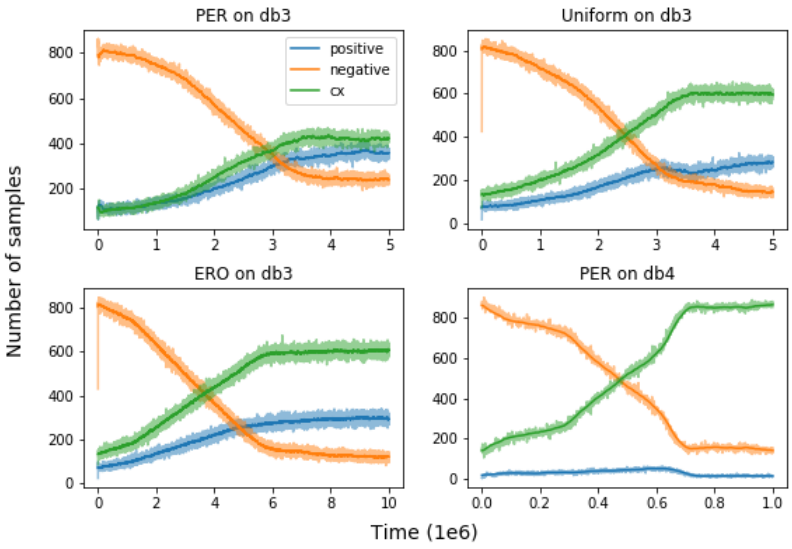


Figure 5.7: Figure showing reward diversity in the sampled minibatch of different ER methods for Set-3 for PER, ER-Uniform, and ERO. Results for PER on Set-4 are also shown.

5.6 Discussion

In this chapter, we proposed a novel experience replay approach, SER, for camera selection where the replay memory is highly imbalanced due to frequently occurring actions. We observed that stratified sampling help to create a diverse minibatch which help learn a better policy in off-policy DQN. We showed that SER applied to DQN outperform other ER methods on various camera network datasets. Our experiments showed that SER based DQN resulted in high recall even in camera networks that have long transition times, thus showing that SER successfully balanced rare and frequent actions while learning a policy for camera selection. We showed that our method makes a very few number of re-identification queries than other camera selection methods.

■

Chapter 6

State Representation Learning Based Camera Selection Decisions

In the previous chapters, we presented a novel camera selection method and proposed a modification in experience replay method that helped to learn a policy in biased action space. In this chapter, we will show that the action history used in the previous method is an important state variable and it influences the policy's decision of which camera to select next. But retaining longer history as one-hot vector is computationally expensive. We will now present a new method using state representation learning for making camera selection decisions. We will also present a modification in the training method which reduces the reliance on frame level annotation data. This modification skips the reward for a few frames and takes a reward for every 5th frame from the ground truth. Using reward accumulation, we provide discounted reward to each step of training and learn a policy using Deep Q network (DQN). This is an important to see how RL can be used to train algorithm with less annotated data. We will

demonstrate the performance of our method on several real datasets.

6.1 Introduction

Camera networks are pervasive and frequently used for various visual analytics applications like video surveillance, crowd behavior analysis, etc. The number of cameras at an airport, train station, malls, etc. has rapidly increased, which makes automated tracking an essential task for visual analytics. These camera networks generate an enormous amount of video data which makes it difficult to process all video frames in real-time.

We have seen that camera selection decision [9] is an effective approach in handling a large number of cameras for enabling target tracking in a camera network. In this chapter, we leverage state representation learning (SRL) to encode the state's history and will show that it enables scalable camera selection decisions in a larger camera network.

Re-identification (Re-ID) and data-association are conventional ways [5, 6] used to associate individual tracklets from different cameras to form the multi-camera trajectory of a particular target. Longer transition times result in more uncertainty about the target's location, necessitating more Re-ID queries and thereby increasing the number of false alarms. For a multi-camera tracking application, false alarms are severely detrimental, as they lead to incorrect target association resulting in tracking an irrelevant target. On the other hand, a false negative from a Re-ID algorithm in a camera frame may not be detrimental so

long as the target is re-identified in one of the subsequent frames of the camera. Therefore, to deal with longer transition times, it is important to decide at every time step whether to make a Re-ID query or not, and if the former, which camera feed(s) to query. As we have seen in the previous chapters, such an *intelligent camera selection* strategy is likely to reduce false alarms at an increased risk of missing the target. It has been shown that reducing redundant querying can benefit the multi-camera tracking performance [7, 8, 9] in both, manual and automated surveillance applications. We further investigate intelligent camera selection and focus on tackling the problem of camera-handovers[1], as we scale to larger camera networks.

In the previous chapters, we showed the efficacy of using DQN for making camera selection decisions. We observe that in the absence of knowledge of the camera topology, the camera history is an important state variable. It holds information about the sequence of previously queried cameras, which influences the decision of which camera to select for the next query. For larger camera networks, retaining longer history of camera selection is necessary to make well-informed camera selection decisions. In this chapter, we argue that hand-crafted state variables may not be representative enough and hinder the scalability of such an approach. Therefore, we instead propose a state representation learning [12] based approach and modify the state-vector accordingly. A representation helps to learn the variations in the environment in a low dimension vector. Our final state vector leverages an LSTM-based autoencoder (AE) to summarize

[1]We will use words camera-transition and camera-handover interchangeably.

the camera history of Re-ID queries. We will also show advantages of using a learned state representation, including generalization across camera-network datasets, accommodating a generic DQN architecture across datasets (unlike [63]), and most importantly reduced training speeds. We will show that the AE trained once on a larger dataset works for all smaller datasets as well.

In this context, our specific contributions are summarized below:

1. We propose a novel method for making camera selection decision using state representation learning (SRL). We employ an LSTM based autoencoder (AE) for latent representation of history vector in the state. We will show empirically that learned state representation, as opposed to handcrafted state variables, achieve state-of-the-art results as train faster.

2. We show benefits achieved using SRL for tracking targets in a camera network. For example, the latent representation helps to use the same network architecture (unlike [63]) to larger camera networks and achieves state-of-the-art in camera selection performance.

3. We use a reward function that helps to reduce the amount of supervision in training the policy and the proposed method can be trained in a semi-supervised manner by discounted reward given after skipping some frames. We will show that this achieves comparable performance with the supervised policy.

4. We will show that our proposed method for camera selection decision benefit real applications like multi-target multi-camera (MTMC) tracking and

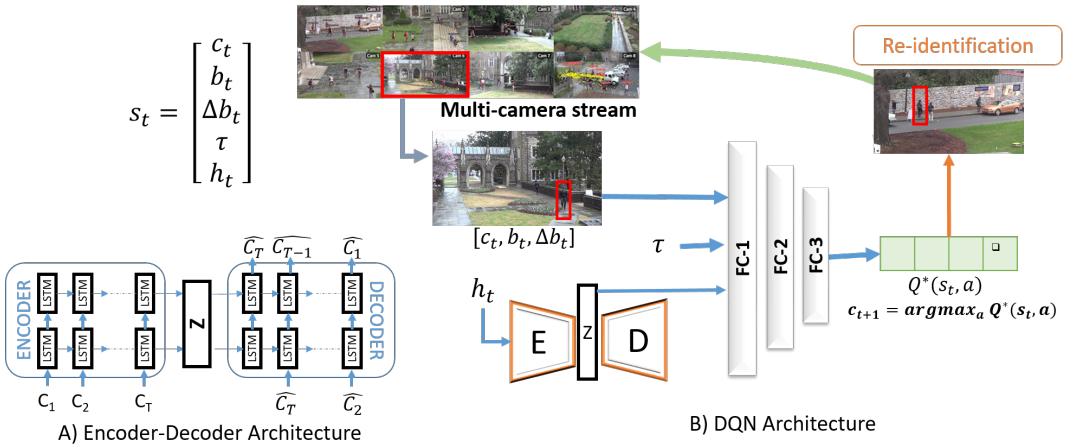


Figure 6.1: A) The architecture of the LSTM based autoencoder that is used to encode the action history in a fixed length latent representation (Z). B) The DQN architecture used to learn the camera selection policy. The neural network model that learns the policy takes as input the different state variables and the action history (h_t) encoded using the LSTM based Autoencoder (E-Encoder, D-Decoder).

multi-camera trajectory forecasting (MCTF) in a camera network.

5. We demonstrate the camera selection performance on four real datasets, NLPR MCT dataset [2], Duke MTMC dataset [5], WNMF dataset [4], and CityFlow dataset [90, 91]. The extensive experiments show that the proposed method is superior than most state-of-the-art methods and is target agnostic.

6.2 Proposed Method

In this section, we will explain the proposed method, the formulation of camera selection decisions using Markov Decision Process (MDP), the neural network model for camera selection policy and its training.

6.2.1 Camera Selection as an MDP

An MDP (Markov Decision Process) is defined as a tuple of elements $(S, \mathcal{A}, f, R, \gamma)$, where S is the state space, $\mathcal{A}$ is the action space, $f(s_t, s_{t+1})$ is the state transition function, $R(s, a)$ is the reward function and γ is the discount factor. We model the camera selection problem as a finite horizon discounted sum reward problem. The state variable has changed from previous Chapter 5 and the individual elements of the MDP are described below:

State: In the camera network, we have access to the location (bounding box) of the target in a given camera frame. The last seen location of the target is represented as (c, b), where c is the camera index (encoded by a one-hot vector) and b is the bounding box (represented as $[x, y, w, h]^\top$). To include the direction of motion of the target, we include the deltas of the bounding box ($\Delta b_t = b_t - b_{t-1}$) in the state vector. As the target is moving across different cameras, it may not be visible at all times, for example, during an occlusion or when it is transitioning between camera FOVs. To handle inter-camera transitions of the target, we include a time-progress variable τ that monitors the number of timesteps elapsed since the last time the target was observed. Additionally, we also maintain a history of past actions (camera selections) as part of the state vector (which is represented as a sequence of one-hot vectors in the previous method in Chapter 5) representing which cameras have been queried by the policy in the last 20 timesteps. The final state is given by the set $s_t = (c, b, \Delta b, \tau, h_t)$. It is worth emphasizing that the camera history in its

raw form is a sequence of one-hot encoded vectors representing the cameras queried, so we use an autoencoder model to learn latent embeddings that are fixed-length representations for this state variable. It also doesn't change the size of state variable in regard to the history vector.

Action: The action space is encoded as $\mathcal{A} = \{0, 1, \ldots, N-1, C_\times\}$, where N is the number of cameras and an action $C_\times$ is included as a 'null camera', suggesting that the target is making an handover and is not visible in the camera network. The policy selects an action $C_\times$ to indicate that no Re-ID queries need to be made.

Reward: We define a reward function for each state action pair. A smaller reward value is taken for the action $C_\times$, which happens to be the most frequent action for ICT. A similar observation is made in [87] to include higher reward for rare transition and lower for frequent.

$$r_{t+1}(s_t, a_t) = \begin{cases} +1 & a_t = y_t \;\&\; \tau > 20 \\ +0.5 & a_t = y_t \;\&\; \tau \leq 20 \\ 0.01 & a_t = y_t = C_\times \\ -1 & \text{otherwise} \end{cases} \tag{6.1}$$

a_t is the action taken and y_t is the ground truth camera. τ is the transition time and it is thresholded to distinguish in-camera occlusions and the camera handovers which are rare and hence a higher reward is provided.

State transition function: With s_t as the state at time t, the policy selects an

action $a_t \in \mathcal{A}$. The next state is updated based on the camera selection action. If the target is found at the selected camera, then the location (c, b) is updated. If not, then τ is incremented and the last policy decision is appended to the history h_t. The latent representation of h_t is used in the state vector.

Q-learning: In reinforcement learning, an agent interacts with its environment by executing an action $a_t \in \mathcal{A}$ at time t by which the environment transitions into the next state s_{t+1} and provides a reward r_{t+1} to the agent. We use Q-value function $Q(s_t, a_t)$ which is the expected discounted sum reward which the agent receives starting from state s_t and taking action a_t at time t. The optimal Q-values $Q^*(s_t, a_t)$ are defined when an optimal policy π^* is followed. Our state-space is continuous and huge and hence we learn a parameterized Q-values $Q^*(s, a|\theta)$ using a neural network.

The optimal Q-values are learned by iteratively updating the parameters θ using deep Q-learning [51] (or also referred to as Deep Q network or DQN). An optimal policy utilizes these Q-values to select an optimal action given the target current state as:

$$\pi_t^*(s_t) = arg\max_a Q^*(s_t, a) \tag{6.2}$$

6.2.2 System Architecture

The proposed architecture is shown in Figure 6.1 to enable target tracking in the camera network. The architecture consists of three important parts. First, the auto-encoder based learned state representation vector, obtained by encoding

the action history into a single fixed length vector. Second, the neural network based policy function, which learns to select a camera given the initial location of the target. Third, the re-identification (Re-ID) algorithm which utilizes the policy-based camera selection to find whether the target is present in the selected camera frame. For this, the current Re-ID features of the target's locations are matched with the template features using threshold based cosine similarity. The threshold and other parameters are explained in the results section. Note that the Re-ID algorithm is external to our work. There are many well-developed existing solutions and we use one such state-of-the-art Re-ID algorithm ABDNet [3].

Learned State Representation. State representation learning (SRL) learns a representation of the agent's state vector into a low dimension vector. This information evolves through time with the actions of the agent in the environment. A representation helps to learn the variations in the environment and the low dimension state-space can be explored faster and overcomes the curse of dimensionality. We found that SRL improves the policy performance for camera selection decisions and speeds up the training process. It also helps to train a policy for larger camera networks (for example, Duke MTMC, CityFlow, and WNMF datasets).

The state variable capturing the past action history of the current policy is an important part of our model. The history encodes previous camera selection decisions that were made, and a long history helps in training the policy. However, with $N + 1$ actions, and the sequence length going in to thousands

for small length sequences makes it difficult to use the history in its raw form (sequence of one-hot encoded vectors). Therefore, we make use of an LSTM-based autoencoder to learn a fixed-vector representation for this sequence of past actions. This architecture is inspired by the success of encoder-decoder architectures [92]. Through our experiments, we find that using such a state representation improves the performance of the policy as well as improves training time. Moreover, we train it on the largest dataset and show that it can be used on all other datasets without any fine-tuning. Refer to Sec. 6.3.2 for a detailed analysis of our state representation learning approach.

Policy Learning. To select a camera, the state vector is constructed based on the initial location, time since observation and the past action history. This state vector is passed through the policy network to select an action (camera index). If the selected action (or camera index) is not $C_{\times}$, then a Re-ID query is made, upon which the Re-ID algorithm determines the presence of the target in the selected camera. If the target is found, then the state is updated with the location (c, b) of the target and history h_t and τ is reset. If the target is absent, then the τ and h_t are updated to obtain the next state vector. This is repeated until the video sequence ends. The resulting reward is then used to train the policy using deep Q-learning [51].

6.2.3 Camera Selection Policy Model

The architecture in Figure 6.1 shows the neural network model which represents the policy for camera selections. The neural network model contains three

Algorithm 5 Training procedure for learning the policy. π is the policy to make camera selection decisions. `c,b` is the initial location of the target with c as the current camera and b as the corresponding bounding box location.

procedure TRAIN(`c,b`, π)
    Initialize replay memory $M_{C_\times}, M_+, M_-$ with capacity D
    Initialize deep-Q network Q with random weights θ
    Initialize target network Q_T with weights $\theta' = \theta$
    $\tau \leftarrow$ `ZEROS` ▷ Initialize time-elapse with zeros
    $h \leftarrow$ `ZEROS` ▷ Initialize history vector of length L with zeros
    $Z_h \leftarrow$ `latentRepresentation(h)`
    $s \leftarrow$ `initialState(c,b,`τ, Z_h`)` ▷ Concatenates location and history
    while True **do**
        `With probability` ϵ, `choose action c uniformly at random, and with probability` $1-\epsilon$, `choose action using the neural network based policy`
        box $\leftarrow$ `getBoundingBox(c)` ▷ get bounding box using re-identification
        if box is NOT EMPTY **then** ▷ if target is re-identified in selected camera
            $x_t \leftarrow$ `(c,box)`
            $\tau \leftarrow$ `ZERO`
        else
            $\tau += 1$
        `h.append(c)`
        $Z_h \leftarrow$ `latentRepresentation(h)`
        $s' \leftarrow$ `f(`x_t, Z_h, τ`)` ▷ observe the next state and reward
        $r \leftarrow$ `getReward(s,c)` ▷ Using equation 6.1
        if $r > 0.1$ **then**
            `Append transition` (s, c, s', r) `to replay memory` M_+
        else if $r < 0$ **then**
            `Append transition` (s, c, s', r) `to replay memory` M_-
        else
            `Append transition` (s, c, s', r) `to replay memory` $M_{C_\times}$
        if s' is terminal **then** break
        $s \leftarrow s'$
    `Sample a random minibatch B equally from` $M_+, M_-, M_{C_\times}$
    `For each sample` (s_i, a_i, s'_i, r_i) `in minibatch, compute target value` $y_i = r_i + \gamma max_a Q_T(s'_i, a)$
    `Update the Q-network using adam algorithm [69] on the minibatch and repeat until convergence`
    return π

fully-connected layers with size (2048,1024,256) and ReLu activation function. The output layer is equal to the number of actions ($N + 1$) with linear activation which represents the Q-value function $Q(s_t, a), \forall a \in \mathcal{A}$. We use the MSE loss to learn the optimal weights for the policy using deep-Q learning. The action history of the policy is represented using LSTM based 2-layer sequence-to-sequence encoder decoder architecture (AE) which is *trained separately* from the policy network. The AE structure is shown in Figure 6.1, and it is trained using the cross-entropy loss. The input given is an action sequence $a_{1:T} = c_1, c_2, \ldots, c_T$, where each c_i is a one-hot encoded vector. The latent vector is the last layer's cell state at time T. The latent vector preserves the information of the sequence, which is used to reconstruct the input using the decoder. The sequence is decoded in the reverse order i.e., $\hat{a}_{1:T} = \hat{c}_T, \hat{c}_{T-1}, \ldots, \hat{c}_1$ and then cross entropy loss is computed between the input sequence and the output sequence (after reverse order taken into account). The loss is back-propagated to learn the optimal weights for the AE. Once trained, the AE can encode the action history into a fixed representation which can directly be used in the state vector to learn the policy. We need not retrain or fine-tune the AE for another dataset, as we established during our experiments that the AE generalizes well across different datasets. The only requirement is that at test time, the number of cameras N_{test}, should be smaller than that at train time N_{train}. We can then encode the past action history by zero-padding the one-hot encoded vector to make it of size N_{train}.

The policy is learned using deep Q-learning with experience replay (ER).

In deep Q-learning, the agent interacts with the environment and executes an action given its state. ER is a technique to store the previous experiences (state-action-reward pair) of the policy to prevent catastrophic forgetting in the neural networks. While back-propagation, these experiences are sampled from the replay buffer to create a minibatch to replay the previous experiences to learn an optimal policy. In case of camera selections, we observed that the experiences corresponding to the action $C_{\times}$ were repeated too many times which creates an imabalanced minibatch and hence biases the policy to the most frequently occurring action. For this, we proposed a new experience replay method (named SER) in the Chapter 5 which creates a diverse minibatch. To create a diverse minibatch, we segregated the replay buffer into three buffers, first, to store the experiences from the most frequent action $C_{\times}$, second, to store the experiences which resulted into a positive reward and third, to store the experiences which resulted in a negative reward. Then we sampled the experiences *uniformly* from all replay buffers to create a diverse and balanced minibatch of all possible experiences. The minibatch is replayed and MSE loss is computed. Let the minibatch be $\mathcal{B} = (s, a, \bar{s}, r)$ which is used to generate an empirical estimate of the expected loss $L(\theta_t)$, as shown in eqn. 6.3

$$L(\theta_t) = \frac{1}{|\mathcal{B}|} \sum_{i=0}^{|\mathcal{B}|} \left[(r_{t+1} + \gamma max_a Q(s_{t+1}, a)) - Q(s_{t+1}, a|\theta_t) \right]^2 \tag{6.3}$$

where $(r_{t+1} + \gamma max_a Q(s_{t+1}, a))$ is the target and $Q(s_{t+1}, a)$ is the output of the neural network.This error is also referred to as TD (temporal-difference) error and is minimized by backpropagation to learn the optimal weights of the policy

network. The training algorithm is shown in algorithm 5.

Semi-supervised Training To train the neural network model without full supervision, the reward function in equation 6.1 is modified keeping other variables same for training. The reward is given after skipping a few frames. For example, if the reward is given after n frames then the discounted reward is accumulated to the previous $n-1$ frames. The discounted reward is computed using discount factor $\gamma = 0.9$. The Figure 6.2 shows the reward given to the neural network during training phase. The figure shows 10 steps of training phase and cameras selected by the policy in the second row. The third row shows the reward given after every 5 frames ($n = 5$) using reward function defined in equation 6.1. The last row shows the reward is discounted for all previous frames where a reward is not given. At any time step t_i, a discounted reward ($\gamma^{n-i}.r$) is given if policy selects same camera as the camera where reward (r) is given otherwise a 0 reward is given. The impact of number of frame-skip is shown in the next section. A Re-ID method is also not used during training when a reward is not given, where a bounding box is picked using intersection-over-union (IOU). A bounding box in the current frame which has 0.6 or more IOU with the previous frame's bounding box is selected for that frame.

6.3 Results

In this section, we will describe the experimental setup, performance evaluation for camera selection and target tracking.

Figure 6.2: Figure shows the modification in the reward function (equation 6.1) for semi-supervised training.

6.3.1 Experimental Setup

Datasets: We have used NLPR-MCT data set [7], DukeMTMC [28], CityFlow [90, 91], and WNMF [4] dataset to evaluate the proposed method for camera selections in multi-camera networks. These datasets are detailed in the Table 6.1. Details of NLPR dataset and DukeMTMC dataset are given in section 3.3.1 of Chapter 3 and section 4.3.1 of Chapter 4. The CityFlow dataset is divided into multiple scenarios, we select two large scenarios (scenario 4 having 25 cameras and scenario 5 having 19 cameras) to demonstrate scalability of our proposed framework for making camera selection decisions. WNMF dataset is consists of 15 cameras and recording is captures for 20 days (600 hours of video footage, but only small tracks of movement are annotation and provided for use). The dataset is collected for trajectory forecasting in a camera network.

Through various experiments, we will show that the proposed method can make camera selection decisions for target types like people/pedestrians (NLPR and DukeMTMC dataset) and vehicles (CityFlow dataset). To demonstrate that

Table 6.1: Details of the datasets used for training and performance evaluation. The table shows the number of cameras (#Cameras), duration of the videos, frame rate (FPS), the number of targets (#Target) captured in each dataset.

	#Cam	Duration	FPS	#Target
NLPR-Set1	3	20 min	20	235
NLPR-Set2	3	20 min	20	255
NLPR-Set3	4	3.5 min	25	14
NLPR-Set4	5	24 min	25	49
DukeMTMC	8	1hr 25min	60	2834
CityFlow S04	25	17.97 mins	10	71
CityFlow S05	19	2hr 3mins	10	337
WNMF	15	600 hrs	5	-

the proposed method scales to larger camera networks, we will show the efficacy of our method on relatively larger datasets (DukeMTMC and CityFlow dataset).

The training set, testing set and evaluation experiments are taken from the state-of-the-art methods [7, 9, 63]. We show performance comparison with various methods through these experiments. A separate policy is trained for each dataset for making camera selection decisions. The DukeMTMC dataset has gone offline due to privacy reasons [93] and hence comparison for tracking according to benchmark cannot be done due to non-availability of the ground truth for computing metric values. Thus, we make a comparison according to the training and testing split provided by [63] for camera selection decisions and tracking. For WNMF dataset, we used first 15 days as the training set and remaining 5 days of recording as testing set.

Performance metric: We evaluate camera selection, inter-camera tracking, and multi-camera multi-target tracking performance separately. To evaluate camera selection performance, we use precision (P), recall (R), and F1 scores [9]. To evaluate the inter-camera tracking and multi-camera tracking performance,

Table 6.2: Camera Selection performance of our proposed method and its comparison with state-of-the-art approaches for NLPR-set1,2,3. The best results are shown in bold and second-best results are italicized.

	NLPR Set-1			NLPR Set-2			NLPR Set-3		
	$P\uparrow$	$R\uparrow$	$F1\uparrow$	$P\uparrow$	$R\uparrow$	$F1\uparrow$	$P\uparrow$	$R\uparrow$	$F1\uparrow$
Exhaustive	0.24	1.0	0.37	0.22	1.0	0.34	0.10	1.0	0.19
Neighbor	0.36	1.0	0.37	0.32	1.0	0.34	0.14	1.0	0.25
CamSel [9]	0.95	0.83	0.86	0.94	0.78	0.82	0.64	0.86	0.72
nStep [63]	0.76	0.90	0.75	0.69	0.84	0.81	0.60	0.88	0.70
Ours	0.92	0.95	0.92	0.92	0.95	0.93	0.68	0.88	0.76
Ours-SemiSup	0.86	0.95	0.89	0.82	0.95	0.86	0.66	0.90	0.75

we use commonly used Multi-Camera Tracking Accuracy (MCTA) metric [7]. The computational performance is quantified using number of spurious frames queried (named metric F). These performance metrics are explained in section 3.3.1 in Chapter 3.

For inter-camera tracking (ICT), we define the measure Percentage Camera Handover (PCH) as the percentage of target transitions (from Camera C_i to C_j, $i \neq j$) that are correctly detected by using the learned policy. Missing more target transitions hurts overall tracking performance, and increases the chance of not finding the target again.

Implementation: We implemented the DQN algorithm using PyTorch framework and utilized a server with 128-GBs of RAM and a 11-GB Nvidia RTX 2080 Ti GPU for training. We used the epsilon-greedy policy as an exploration strategy during training. To train the model, we employ a deep neural network with three hidden layers (4096-1024-256) with *Adam* optimizer. The subsequent sections present the performance on the best parameters.

Table 6.3: Camera Selection performance of our proposed method and its comparison with state-of-the-art approaches on NLPR-Set4 and DukeMTMC dataset. The best results are shown in bold and second-best results are italicized.

	NLPR Set-4			Duke MTMC		
	$P\uparrow$	$R\uparrow$	$F1\uparrow$	$P\uparrow$	$R\uparrow$	$F1\uparrow$
Exhaustive	0.11	1.0	0.19	0.042	1.0	0.11
Neighbor	0.18	1.0	0.29	0.042	1.0	0.33
CamSel [9]	0.61	0.91	0.66	Out of Memory		
nStep [63]	0.73	0.84	0.78	0.49	0.768	0.546
Ours	0.72	0.76	0.71	0.91	0.92	0.91
Ours-SemiSup	0.62	0.84	0.68	0.87	0.96	0.90

6.3.2 Camera Selection Decisions

6.3.2.1 Impact of State-Representation on Performance

In this experiment, we study the state-representation in detail by observing the impact of sequence length and auto-encoder on the camera selection performance. For this, we generate all possible sequences of length $10, 20$, and 50 on various datasets to train the LSTM based encoder-decoder architecture. We observed that AE trained on the sequences of larger dataset (CityFlow with 40 cameras) can also encode the sequences of the smaller datasets (DukeMTMC and NLPR). Hence, we train the AE only once on CityFlow dataset and use it for all the experiments. For this, we generated sequences from CityFlow dataset

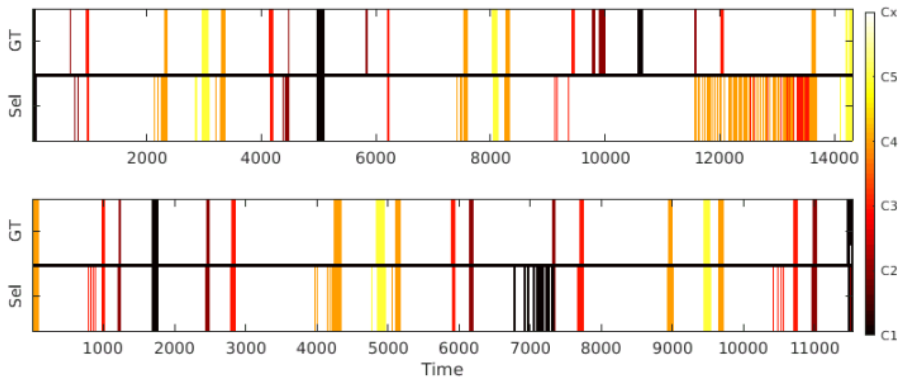


Figure 6.3: Figure shows the qualitative performance of the proposed camera selection method for two targets.

Table 6.4: Testing loss on different datasets with different LSTM size. The corresponding epoch number is indicated in brackets.

Size	NLPR Set-4	DukeMTMC	CityFlow
64	0.041 (5000)	0.098 (621)	0.001 (6586)
128	0.045 (2000)	0.062 (76)	0.001 (270)
256	0.041 (400)	0.008 (130)	0.001 (140)

with epsilon-greedy exploration [94]. To use AE on smaller datasets trained on CityFlow, zeors are padded to the one-hot vector representing a camera index.

To evaluate the efficacy of AE, the generated sequences are divided into training and testing sets by selecting random half in each set. Table 6.4 shows the impact of LSTM node size to encode the camera history. We evaluated for three node sizes 64, 128, and 256. The loss value is the mean cross entropy value for the testing set. The loss value didn't improve beyond 0.001 for CityFlow and hence we use an encoding length of 256. The AE trained on CityFlow is used for all datasets in further experiments for camera selection and tracking.

Table 6.5 shows the impact of sequence length on the camera selection performance on NLPR-Set4. We quantify the impact in terms of PCH metric on the testing set and the number of episodes required to train the policy. We created multiple configurations like training the policy without AE (no AE in the table), using AE trained on the training set of the same dataset which is named as AE(same), finally using AE which is trained on a larger dataset named as AE(Num), where Num is the sequence length of that particular configuration. In the table, we show the number of episodes (in RL, all states between initial and terminal state is one episode, for example, one game of chess) the policy trained using a particular configuration and the corresponding PCH on the testing set.

Table 6.5: Percentage Camera Handovers (PCH) (higher is better) on NLPR-Set4 when trained without AE and with AE. AE(same) represents AE is trained on same dataset, AE (N) represents that AE is trained on a bigger dataset with sequence length N.

Configuration	Episodes	PCH↑
Without AE	25587	53.6
AE (same)	21704	65
AE (10)	25588	62.4
AE (20)	24883	64
AE (50)	25549	64.8

We choose final sequence length to be 20 for all further experiments. This is because PCH for sequence length 20 and 50 is very close but recall for length 20 (78%) is higher than length 50 (74%). PCH without AE didn't improve beyond 53.6 but using AE it reached significant high value in approx. same number of episodes. This means that using state-representation learning not only provides better PCH but also helps in learning the policy faster. Next, we will show the camera selection performance on various datasets and will show that SRL achieves better camera selection performance on relatively larger datasets (CityFlow dataset).

6.3.2.2 Camera Selection Performance

We first evaluate the performance of our camera selection policy in terms of Precision (P), Recall (R), F1-score (F1) metrics. For this experiment, we use the initial location of the target to make the initial state and AE is initialized with a sequence of all zeros. The proposed policy selects a camera from the initial state and then if the target is found in the selected camera then the state is updated accordingly using the state-transition function. The selected camera is appended in the action history and a representation is taken from AE for next decision. For

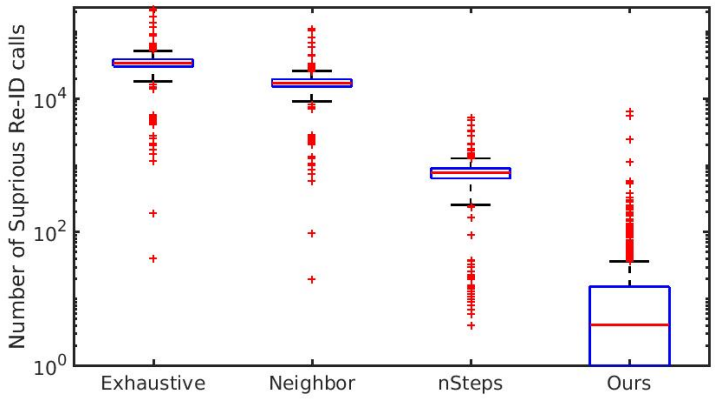


Figure 6.4: The re-identification calls made by different methods on DukeMTMC dataset.

this experiment, the target is re-identified using ground truth to test the camera selection decisions alone. The camera selection performance is shown in the Table 6.2 and 6.3. We compare the performance with state-of-the-art methods [9] and [63] and other baseline methods *Neighbor*, *Exhaustive*. *Exhaustive* queries all cameras at all times. *Neighbor* assumes that the camera network topology is known and queries only the neighboring cameras. As expected, the *Neighbor* and *Exhaustive* have perfect recall (please note that ground truth is used for re-identification in this experiment) but poor precision and the false alarms are detrimental to the performance if there are errors in re-identification. Hence, not selecting a camera at times will be beneficial (see section6.3.4). Our proposed policy performs better on various cases as shown in the table. The Figure 6.4 shows the number of re-identification calls made by different methods and our proposed policy makes very fewer calls as compared to the related and baseline methods. CamSel [9] goes out of memory on DukeMTMC dataset and hence the results are not reported in the figure.

The qualitative performance for camera selections is shown in Figure 6.3 for

NLPR-Set4 for two targets. The figure shows the number transitions captured by different methods where each color depicts a particular camera FOV and the white color depicts the target is transition between two cameras. For each target, there are two sequences, *Sel* and *GT*. The top is the ground truth sequence of cameras (*GT*) and lower is the selected sequence of cameras (*Sel*). The figure shows that the policy helps to capture all the camera handovers made by the targets. The Figure 6.5 shows the PCH captured by our method and nSteps [63] which is the state-of-the-art camera selection method on these dataset. PCH is computed as the percentage of target transitions (from Camera C_i to C_j, $i \neq j$) that are correctly detected by using the learned policy. Missing more target transitions hurts overall tracking performance, and increased chances of not finding the target again. The figure shows that our proposed method leads to an absolute improvement of 39% for NLPR-Set4 and 35% for DukeMTMC datasets over nSteps method. This increase is substantial for NLPR-Set4 and DukeMTMC that have higher target transition times.

6.3.2.3 Semi-supervised Training

To show that camera selection policy can be trained with limited supervision, we provide reward after skipping a few frames as explained in section 6.2. In this experiment, we will first show the impact of number of frames skipped before reward is given for NLPR-Set4 and then the policy's performance on all datasets.

Table 6.6 shows the Precision (P), Recall (R), F1-scores (F1), and Percentage

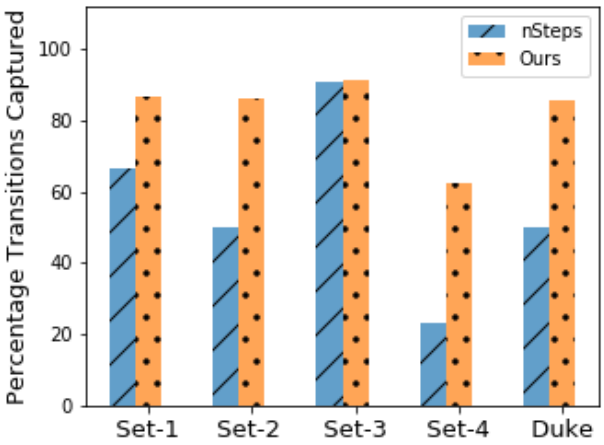


Figure 6.5: Figure compares the Percentage Camera Handover (PCH) of our method with the state-of-the-art method.

Table 6.6: Camera selection performance for semi-supervised training on NLPR Set-4.

FPS (frame-skip)	P↑	R↑	F1↑	PCH↑
0.5 (20)	0.498	0.569	0.507	0.6
1 (10)	0.594	0.752	0.634	0.592
2 (5)	0.621	0.839	0.677	0.736
5 (2)	0.67	0.84	0.72	0.728

camera handover (PCH) for different number of frames skipped before providing reward. Finally, a frame-skip value of 5 is used for all datasets. By using a reward every 5 frames, the annotation cost is reduced by 5 times. The performance on all datasets is shown in Table 6.2 and 6.3.

6.3.3 Scalable Camera Selection Decisions

In this subsection, we will show experiments which show that the proposed method is scalable to larger camera networks. Figure 6.6 shows the learned topology of the DukeMTMC dataset as a colormap. Both axes are camera numbers which shows whether a link $c_i - c_j$ exist between the two cameras indices. The color intensity shows the number of transitions present in ground truth and

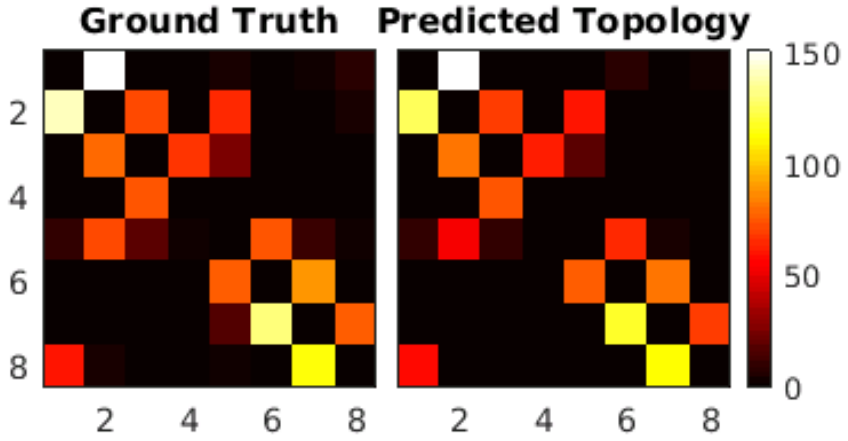


Figure 6.6: Ground Truth topology and predicted topology of the DukeMTMC dataset. Both axes show the camera index and color intensity represents the number of transitions in a $c_i - c_j$ transition.

Table 6.7: The camera selection performance on two scenarios of the CityFlow dataset.

	Scenario 5			Scenario 4		
	P↑	R↑	F1↑	P↑	R↑	F1↑
Neighbor	0.32	1.0	0.49	0.22	1.0	0.36
nsteps	0.40	0.42	0.40	0.45	0.52	0.48
Ours	0.82	0.87	0.81	0.84	0.92	0.84

in the predicted trajectories. The PCH for Duke dataset is shown in Figure 6.5. Above results indicates that our proposed method achieves state-of-the-art on DukeMTMC dataset.

The camera selection performance for CityFlow dataset is shown in Table 6.7 that shows significantly larger values for all performance metrics. We do not include comparison on CityFlow dataset as no benchmark algorithm exists for this dataset for camera selections.

6.3.4 Benefits of Camera Selection Decision in a Camera Network

In this subsection, we include two experiment to show how camera selection decision benefit different applications. We will show benefits in multi-target multi-

camera tracking (MTMC), and multi-camera trajectory forecasting (MCTF).

Table 6.8: Average MCTA values (higher is better) for ICT alone case on NLPR-MCT and DukeMTMC dataset. The results are separated based on the type of association method. *Self* means a method uses its own association, *GT* represents ground truth, and *Re-ID* signifies that a Re-ID method is used for association. We used ABDNet [3] for Re-ID.

		Inter-camera tracking (ICT)				
Approach	Association	Set-1	Set-2	Set-3	Set-4	Duke
[10]	Self	0.9152	0.9132	0.5163	0.7152	-
[2]	Self	0.7425	0.6544	0.7369	0.3945	-
[64]	Self	0.6617	0.5907	0.7105	0.5703	-
[38]	Self	0.3203	0.3456	0.1381	0.1562	-
[7]	Self	0.9610	0.9264	0.7889	0.7578	-
[1]	Self	0.835	0.703	0.742	0.385	-
CamSel [9]	GT	0.8210	0.7498	0.9099	0.8993	Mem
nSteps [63]	GT	0.9016	0.8741	0.9038	0.8074	0.8027
Ours	GT	0.968	0.963	0.914	0.759	0.902
Neighbor	Re-ID	0.6405	0.3627	0.2618	0.5386	0.6784
Ours	Re-ID	0.9292	0.8806	0.8426	0.7808	0.8855

Table 6.9: Average MCTA values (higher is better) for both SCT-ICT case on NLPR-MCT and DukeMTMC dataset. The results are separated based on the type of association method. *Self* means a method uses its own association, *GT* represents ground truth, and *Re-ID* signifies that a Re-ID method is used for association. We used ABDNet [3] for Re-ID.

	Single-camera tracking + ICT					
Approach	Association	Set-1	Set-2	Set-3	Set-4	Duke
[10]	Self	0.8831	0.8397	0.2427	0.4357	-
[2]	Self	0.7477	0.6561	0.2028	0.2650	-
[64]	Self	0.6903	0.6238	0.0848	0.1830	-
[38]	Self	0.8162	0.7730	0.1240	0.4637	-
[1]	Self	0.8525	0.7370	0.4724	0.3778	-
CamSel [9]	GT	0.8235	0.7503	0.9134	0.9118	Mem
nSteps [63]	GT	0.9018	0.8806	0.9058	0.7871	0.8191
Ours	GT	0.966	0.961	0.906	0.776	0.894
Neighbor	Re-ID	0.5119	0.2564	0.1445	0.4426	0.5487
Ours	Re-ID	0.7639	0.7594	0.3547	0.5258	0.7308

6.3.4.1 Multi-Camera Target Tracking

In this experiment, we will show the tracking performance while tracking the target using the learned policy for camera selections. We will also compare the tracking performance with state-of-the-art methods on NLPR and DukeMTMC

datasets using MCTA metric. For this experiment, the initial position of a target and the state representation of zero-initialized action history are used to make the initial state. The initial state is then used by the learned policy to select a camera where the target is expected to reappear at the next time instant. If the target is present in the selected camera frame then the location of the target is updated accordingly in the state vector otherwise the time-elapse (τ) is updated. The procedure is repeated until the video sequence ends. For re-identification, we have used pre-trained model of ABDNet [3] for DukeMTMC dataset. We used same model for all datasets of NLPR to avoid re-training for re-identification. Please note that our method is single target multi-camera tracking approach and to make it work for multiple targets, we run multiple parallel pipeline of our method starting from the initial location of the target.

Camera selections inherently improves the tracking performance as shown in the Table 6.8 and 6.9 which shows tracking performance on NLPR and DukeMTMC dataset using MCTA metric. The methods in the table are separated based on how these methods resolve the camera handover (re-identifying the target). In the tables (6.8 and 6.9), *Self* means that the methods have proposed their own approach to resolve the camera handover, *GT* signifies that methods use ground truth for resolving the handover, and *Re-ID* means that a re-identification method in [3] is used. There are two experiments as used in the literature [7, 38, 64, 1, 10]. In experiment-1, only the inter-camera tracking (ICT) performance is evaluated. For this, detection and single camera tracking are taken from the ground truth. The camera selection decisions are taken at

Table 6.10: The camera selection performance on WNMF dataset. The baseline methods are taken from dataset baselines [4] and LSTM (Cam. Sel.) is a camera selection based baseline.

Model	Accuracy(%) Top 1↑	Top 3↑
Shortest real-world distance	46.8	92.2
Most frequent transition	65.7	91.8
Most similar trajectory	69.7	94.5
Hand-crafted features	70.7	94.1
Fully-connected network	73.4	95.1
LSTM (Pred.)	74.4	94.2
GRU (Pred.)	75.1	94.9
Ours (Pred.)	79	94
LSTM (Cam. Sel.)	63.1	91.5
Ours (Cam. Sel.)	93.28	96.27

all times during ICT and a re-identification query is made when a non-$C_{\times}$ camera is selected. In experiment-2 (ICT+SCT), only the detections are taken from ground truth. The policy is used at all times both when the target is transitioning and moving in a particular camera FOV. A re-identification query is resolved accordingly using ABDNet [3]. The table shows that our method is better on most of the datasets especially on the datasets that have higher number of cameras. The approach *Neighbor* is a baseline method where we assume that the camera topology is known and the neighboring cameras are queried to resolve the camera handover, our policy achieves better performance than this baseline method on all cases and hence not selecting cameras during camera handover improves the tracking performance. The dashed values in the table means that a method doesn't report those results.

6.3.4.2 Multi-Camera Trajectory Forecasting

In this section, we will be showing an additional application where our framework can be used. Multi-Camera Trajectory Forecasting (MCTF) is a task where the future trajectory of an object is predicted in a network of cameras [4]. An effective MCTF model should proactively anticipate where and when a person will re-appear in the camera network after departure from another camera in the same camera network. This requires the MCTF model to identify the next camera where the target will re-appear, the transition time, and the location in the identified camera.

Our original framework is for making camera selection decisions and not MCTF but we claim that our framework can be used for MCTF to predict the next location in a sequential manner. The length of the camera selected as $c_\times$ is the transition time, and the camera selected as non-$c_\times$ is the next camera (where the target is re-identified). The location in the next camera will be taken care by a re-identification method [37]. Our framework is used in the same manner as used in target tracking in a camera network in section 6.3.4.1. The camera selection policy gives the next camera where the target will re-appear, the length of $C_\times$ selection gives the transition time, and the re-identification block gives the location of the target in next camera.

We compare the performance with several baseline methods as described in [4]. These baselines focus only on the next camera prediction and ignore the transition time. Hence, we introduce another baseline method where we use an

LSTM based approach for making camera selection decisions in a sequential manner as compared to direct prediction. This baseline is trained as an encoder-decoder using supervised learning. It selects a camera each timestep and if the selected camera is $c_{\times}$ then nothing changes otherwise the bounding box location of the target is picked using a Re-ID method. This baseline is named LSTM (cam. sel.).

The performance comparison of these baselines with our method is shown in Table 6.10. The table shows top-1 and top-3 accuracy for next camera prediction. The table shows all baselines from the dataset results [4] and our methods. LSTM (pred.) is a method which predicts the next camera using an LSTM, Ours (Pred.) is our method used for prediction which achieves 4% better top-1 accuracy than other baselines in prediction. In this, the first non-$c_{\times}$ camera is used as the next camera from the sequence of selected cameras. Whereas, when our method is used as a selection framework (Ours (cam. sel.) in table) then its top-1 accuracy is 18% more than the second best baseline.

Figure 6.7 shows the camera transition for 5 targets in the WNMF dataset. In the figure, the ground truth and selected camera sequences for one target are shown in the same plot. Each color represents a camera index and white space represents the transition time. Colorbar and y-axis gives information of the camera index and ground truth sequence respectively. The first plot in the figure show a little occlusion in the sequence of cameras and the learned policy effectively works during that period as well. For all other targets, the next camera is well identified and corrected when a wrong decision is made (target 1 and

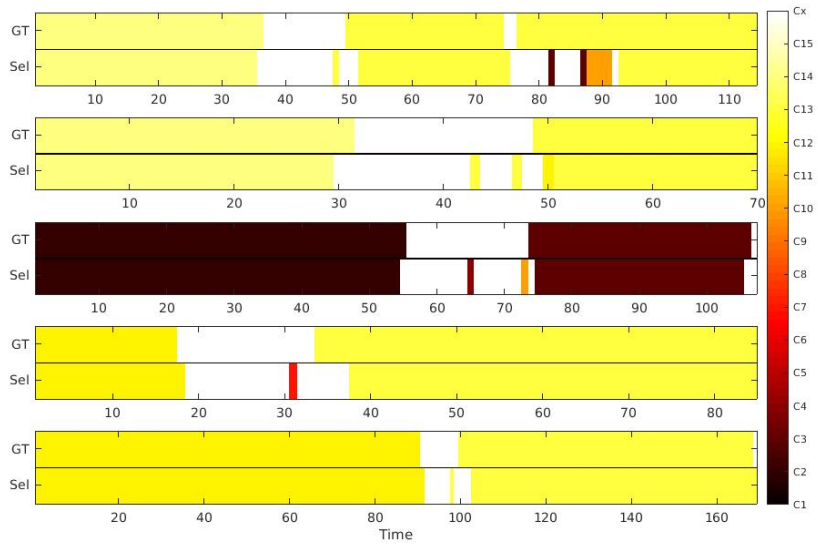


Figure 6.7: Figure shows the difference in the transition time captured of multiple tracks/targets of WNMF dataset.

4 in the figure).

Another important task to perform in MCTF is identifying the transition time. The total time-steps when $c_{\times}$ is selected by the policy or till when the target is not re-identified is the transition time for our method. This is shown in Figure 6.8. It shows the ground truth (oval markers) and predicted transition time (cross marker) for a few good and bad tracks (or targets). The difference in transition time is represented by vertical lines. Overall, 80% of the tracks have a difference of less than 10 frames.

6.4 Discussion

In this chapter, we proposed a method to make camera selection decisions using state representation learning and DQN approach in reinforcement learning. We encoded the action history using LSTM based Auto-Encoder (AE) that

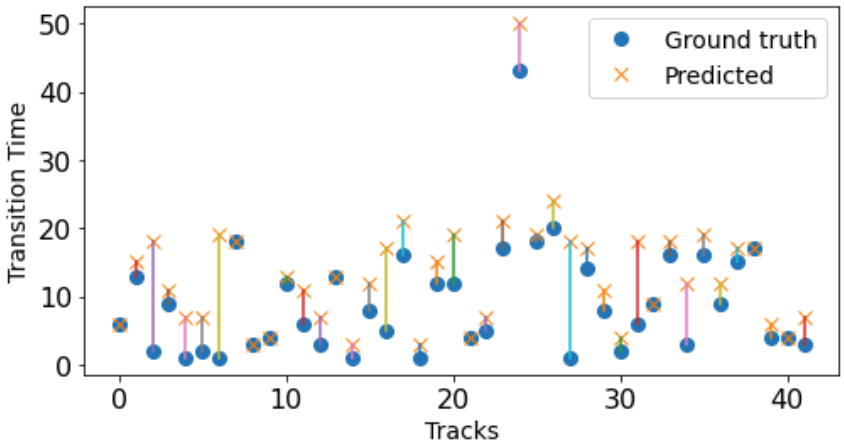


Figure 6.8: Figure shows the difference in the transition time captured of multiple tracks/targets of WNMF dataset.

helped to learn the policy faster as compared to one-hot encoding. We showed that making fewer re-identification queries are crucial for tracking performance. Through various other experiments on NLPR and DukeMTMC datasets, we also showed that our policy achieves better camera selection performance and better tracking performance than various state-of-the-art methods. Later, we showed that our method scales to larger camera networks such as DukeMTMC and CityFlow dataset for making camera selection decisions.

■

Chapter 7

Conclusion and Future Work

The number of cameras at an airport, railway station, malls, etc. has rapidly increased and they produce a deluge of data. In this dissertation, we proposed several methods to make camera selection decisions using DQN approach in reinforcement learning. The proposed methods reduces the number of re-identification queries required to enable automated tracking in a camera network. We also proposed a stratified sampling based experience replay method that help to create a diverse minibatch for DQN. We showed that making fewer re-identification queries are crucial for tracking performance. Through various other experiments on NLPR and DukeMTMC datasets, we also showed that our policy achieves better camera selection performance and better tracking performance than various state-of-the-art methods. Later, we showed that our method scales to larger camera networks such as DukeMTMC and CityFlow dataset for making camera selection decisions.

We used state-representation learning with DQN which helps to train the

policy faster and easily scales to larger camera networks. We also showed a semi-supervised training of DQN which achieved comparable performance than the supervised training of DQN. We showed that using camera selections benefit several applications in a camera network such as multi-target multi-camera (MTMC) tracking and multi-camera trajectory forecasting (MCTF).

In this dissertation, we addressed the camera selection tasks that are crucial to effectively use the multiple video streams of a camera network. We also suggest the possible future extensions of our work. We validated our work on several datasets and showed that the proposed approaches are target agnostic. However, for generalization and to minimize the training overhead on a new camera network, transfer learning based method can be explored. This will also be beneficial when the camera network topology changes by adding or removing a camera from the camera network. To facilitate this, one may also use the camera network topology as a prior and explore similar transitions made by other targets in the network.

■

References

[1] L. C. W. Chen, X. Chen, K. Huang, K. Huang, and K. Huang, "An equalized global graph model-based approach for multicamera object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 11, pp. 2367–2381, Nov 2017.

[2] W. Chen, X. Chen, and K. Huang, "Multi-Camera Object Tracking (MCT) Challenge," http://http://mct.idealtest.org/Datasets.html/, 2014.

[3] T. Chen, S. Ding, J. Xie, Y. Yuan, W. Chen, Y. Yang, Z. Ren, and Z. Wang, "Abd-net: Attentive but diverse person re-identification," 2019.

[4] O. Styles, T. Guha, V. Sanchez, and A. Kot, "Multi-camera trajectory forecasting: Pedestrian trajectory prediction in a network of cameras," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 4379–4382.

[5] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] N. Jiang, S. Bai, Y. Xu, C. Xing, Z. Zhou, and W. Wu, "Online inter-camera trajectory association exploiting person re-identification and camera topology," in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1457–1465. [Online]. Available: https://doi.org/10.1145/3240508.3240663

[7] Y. Lee, Z. Tang, J. Hwang, and Y., "Online-learning-based human tracking across non-overlapping cameras," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2870–2883, Oct 2018.

[8] O. Javed, K. Shafique, Z. Rasheed, and M. Shah, "Modeling inter-camera space-time and appearance relationships for tracking across non-overlapping views," *Comput. Vis. Image Underst.*, vol. 109, no. 2, pp. 146–162, Feb. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.01.003

[9] A. Sharma, S. Anand, and S. Kaul, "Reinforcement learning based querying in camera networks for efficient target tracking," in *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*, 07 2019.

[10] G. M. Y. Cai, "Exploring context information for inter-camera multiple target tracking," in *IEEE Winter Conference on Applications of Computer Vision*, March 2014, pp. 761–768.

[11] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 4045–4054. [Online]. Available: http://proceedings.mlr.press/v80/pardo18a.html

[12] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Integrating state representation learning into deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1394–1401, 2018.

[13] R. Hamid, R. K. Kumar, M. Grundmann, K. Kim, I. Essa, and J. Hodgins, "Player localization using multiple static cameras for sports visualization," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 731–738.

[14] S. Zhang, Y. Zhu, and A. Roy-Chowdhury, "Tracking multiple interacting targets in a camera network," *Computer Vision and Image Understanding*, vol. 134, pp. 64 – 73, 2015, image Understanding for Real-world Distributed Video Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314215000168

[15] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1355–1360, Oct 2003.

[16] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps, "Dynamic subspace-based coordinated multicamera tracking," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2462–2469.

[17] M. Bredereck, X. Jiang, M. Körner, and J. Denzler, "Data association for multi-object tracking-by-detection in multi-camera networks," in *2012 Sixth International Conference on Distributed Smart Cameras (ICDSC)*, Oct 2012, pp. 1–6.

[18] S. Zhang, E. Staudt, T. Faltemier, and A. K. Roy-Chowdhury, "A camera network tracking (camnet) dataset and performance baseline," in *2015 IEEE Winter Conference on Applications of Computer Vision*, Jan 2015, pp. 365–372.

[19] C.-H. Kuo, C. Huang, and R. Nevatia, "Inter-camera association of multi-target tracks by on-line learned appearance affinity models," in *Proceedings of the 11th European Conference on Computer Vision Part I*, ser. ECCV2010. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 383–396. [Online]. Available: http://dl.acm.org/citation.cfm?id=1886063.1886093

[20] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, June 2004, pp. II–205–II–210 Vol.2.

[21] X. Chen, L. An, and B. Bhanu, "Multitarget tracking in nonoverlapping cameras using a reference set," *IEEE Sensors Journal*, vol. 15, no. 5, pp.

2692–2704, May 2015.

[22] S. Daliyot and N. S. Netanyahu, "A framework for inter-camera association of multi-target trajectories by invariant target models," in *Computer Vision - ACCV 2012 Workshops*, J.-I. Park and J. Kim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 372–386.

[23] K. W. Chen, C. C. Lai, P. J. Lee, C. S. Chen, and Y. P. Hung, "Adaptive learning for target tracking and true linking discovering across multiple non-overlapping cameras," *IEEE Transactions on Multimedia*, vol. 13, no. 4, pp. 625–638, Aug 2011.

[24] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera people tracking with a probabilistic occupancy map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 267–282, Feb 2008.

[25] J. Wan and Liu Li, "Distributed optimization for global data association in non-overlapping camera networks," in *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, Oct 2013, pp. 1–7.

[26] Li Zhang, Yuan Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[27] E. Ristani and C. Tomasi, "Tracking multiple people online and in real time," in *Computer Vision – ACCV 2014*, D. Cremers, I. Reid, H. Saito, and M.-H. Yang, Eds. Cham: Springer International Publishing, 2015, pp. 444–459.

[28] 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.

[29] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based multiple-person tracking with partial occlusion handling," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 1815–1821.

[30] Bo Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1, Oct 2005, pp. 90–97 Vol. 1.

[31] V. Chari, S. Lacoste-Julien, I. Laptev, and J. Sivic, "On pairwise costs for network flow multi-object tracking," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5537–5545, 2015.

[32] R. T. Collins, "Multitarget data association with higher-order motion models," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 1744–1751.

[33] A. Das, A. Chakraborty, and A. K. Roy-Chowdhury, "Consistent re-identification in a camera network," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 330–345.

[34] A. Dehghan, S. M. Assari, and M. Shah, "Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 4091–4099.

[35] Shafique and Shah, "A non-iterative greedy algorithm for multi-frame point correspondence," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 110–115 vol.1.

[36] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 3701–3710.

[37] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi, "Deep learning for person re-identification: A survey and outlook," 2020.

[38] X. Chen and B. Bhanu, "Integrating social grouping for multitarget tracking across cameras in a crf model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 11, pp. 2382–2394, Nov 2017.

[39] S. Sunderrajan and B. S. Manjunath, "Multiple view discriminative appearance modeling with imcmc for distributed tracking," in *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, Oct 2013, pp. 1–7.

[40] A. Gilbert and R. Bowden, "Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity," in

Computer Vision – ECCV 2006, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 125–136.

[41] T. D'Orazio, P. L. Mazzeo, and P. Spagnolo, "Color brightness transfer function evaluation for non overlapping multi camera tracking," in *2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, Aug 2009, pp. 1–6.

[42] B. Prosser, S. Gong, and T. Xiang, "Multi-camera matching using bi-directional cumulative brightness transfer functions," in *Proc. BMVC*, 2008, pp. 64.1–64.10, doi:10.5244/C.22.64.

[43] T. Huang and S. Russell, "Object identification in a bayesian context," in *In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97*. Morgan Kaufmann, 1997, pp. 1276–1283.

[44] H. Pasula, S. Russell, M. Ostland, and Y. Ritov, "Tracking many objects with many sensors," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1160–1167. [Online]. Available: http://dl.acm.org/citation.cfm?id=1624312.1624384

[45] Y. T. Tesfaye, E. Zemene, A. Prati, M. Pelillo, and M. Shah, "Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets," *CoRR*, vol. abs/1706.06196, 2017. [Online]. Available: http://arxiv.org/abs/1706.06196

[46] K. Yoon, Y. Song, and M. Jeon, "Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views," *IET Image Processing*, vol. 12, no. 7, pp. 1175–1184, 2018.

[47] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 4705–4713.

[48] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1349–1358.

[49] L. Paletta, G. Fritz, and C. Seifert, "Q-learning of sequential attention for visual object recognition from informative local descriptors," in *Proceedings of the 22Nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 649–656. [Online]. Available: http://doi.acm.org/10.1145/1102351.1102433

[50] S. Karayev, M. Fritz, and T. Darrell, "Anytime recognition of objects and scenes," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 572–579.

[51] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[52] A. Sharma and A. B. Buduru, "Foresee: Attentive future projections of chaotic road environments," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 2073–2075.

[53] A. Sharma and P. Kumar, "Foresee: Attentive future projections of chaotic road environments with online training," *CoRR*, vol. abs/1805.11861, 2018. [Online]. Available: http://arxiv.org/abs/1805.11861

[54] A. Sharma, "Intelligent querying in camera networks for efficient target tracking," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 6458–6459.

[55] M. K. Pal, R. Bhati, A. Sharma, S. K. Kaul, S. Anand, and P. B. Sujit, "A reinforcement learning approach to jointly adapt vehicular communications and planning for optimized driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3287–3293.

[56] R. BELLMAN, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957. [Online]. Available: http://www.jstor.org/stable/24900506

[57] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[58] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2894–2902.

[59] J. S. S. III and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," *CoRR*, vol. abs/1707.04991, 2017. [Online]. Available: http://arxiv.org/abs/1707.04991

[60] W. Luo, P. Sun, Y. Mu, and W. Liu, "End-to-end active object tracking via reinforcement learning," *CoRR*, vol. abs/1705.10561, 2017. [Online]. Available: http://arxiv.org/abs/1705.10561

[61] A. Sharma, M. K. Pal, S. Anand, and S. K. Kaul, "Stratified sampling based experience replay for efficient camera selection decisions," in *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, 2020, pp. 144–151.

[62] A. Sharma, S. Anand, and S. K. Kaul, "Intelligent camera selection decisions for target tracking in a camera network," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 3083–3092.

[63] ——, "Intelligent querying for target tracking in camera networks using deep q-learning with n-step bootstrapping," *Image and Vision Computing*, 2020.

[64] W. Chen, L. Cao, X. Chen, and K. Huang, "A novel solution for multi-camera object tracking," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 2329–2333.

[65] Z. Zheng, L. Zheng, and Y. Yang, "A discriminatively learned cnn embedding for person reidentification," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 1, pp. 13:1–13:20, Dec. 2017. [Online]. Available: http://doi.acm.org/10.1145/3159171

[66] B. Lavi, M. F. Serj, and I. Ullah, "Survey on deep learning techniques for person re-identification task," *CoRR*, vol. abs/1807.05284, 2018. [Online]. Available: http://arxiv.org/abs/1807.05284

[67] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[68] M. J. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *CoRR*, vol. abs/1507.06527, 2015. [Online]. Available: http://arxiv.org/abs/1507.06527

[69] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[70] *MATLAB version 9.10.0.1613233 (R2016b)*, The Mathworks, Inc., Natick, Massachusetts, 2016.

[71] Y. T. Tesfaye, E. Zemene, A. Prati, M. Pelillo, and M. Shah, "Multi-target tracking in multiple non-overlapping cameras using fast-constrained dominant sets," *International Journal of Computer Vision*, vol. 127, no. 9, pp. 1303–1320, 2019.

[72] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[73] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3381–3388.

[74] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

[75] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in *NIPS'16 Workshop on Learning, Inference and Control of Multi-Agent Systems*, Dec. 2016. [Online]. Available: https://sites.google.com/site/malicnips2016/papers

[76] Y. Pan, M. Zaheer, A. White, A. Patterson, and M. White, "Organizing experience: a deeper look at replay mechanisms for sample-based planning in continuous state domains," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018*, 7 2018, pp. 4794–4800. [Online]. Available: https://doi.org/10.24963/ijcai.2018/666

[77] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *International Conference on Learning Representations*, Puerto Rico, 2016.

[78] D. Zha, K.-H. Lai, K. Zhou, and X. Hu, "Experience replay optimization," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 7 2019, pp. 4243–4249. [Online]. Available: https://doi.org/10.24963/ijcai.2019/589

[79] M. Fang, C. Zhou, B. Shi, B. Gong, W. Xi, T. Wang, J. Xu, and T. Zhang, "DHER: Hindsight experience replay for dynamic goals," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Byf5-30qFX

[80] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, p. 27, 2019.

[81] S. Pouyanfar, Y. Tao, A. Mohan, H. Tian, A. S. Kaseb, K. Gauen, R. Dailey, S. Aghajanzadeh, Y. Lu, S. Chen, and M. Shyu, "Dynamic sampling in convolutional neural networks for imbalanced data classification," in *2018*

IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), April 2018, pp. 112–117.

[82] G. Novati and P. Koumoutsakos, "Remember and forget for experience replay," *CoRR*, vol. abs/1807.05827, 2018. [Online]. Available: http://arxiv.org/abs/1807.05827

[83] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 1146–1155. [Online]. Available: http://proceedings.mlr.press/v70/foerster17b.html

[84] R. Zhao, X. Sun, and V. Tresp, "Maximum entropy-regularized multi-goal reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 7553–7562. [Online]. Available: http://proceedings.mlr.press/v97/zhao19d.html

[85] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[86] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1928–1937. [Online]. Available: http://proceedings.mlr.press/v48/mniha16.html

[87] E. Lin, Q. Chen, and X. Qi, "Deep reinforcement learning for imbalanced classification," *CoRR*, vol. abs/1901.01379, 2019. [Online]. Available: http://arxiv.org/abs/1901.01379

[88] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[89] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, and I. Stoica, "Ray rllib: A composable and scalable reinforcement learning library," *arXiv preprint arXiv:1712.09381*, 2017.

[90] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[91] M. Naphade, Z. Tang, M.-C. Chang, D. C. Anastasiu, A. Sharma, R. Chellappa, S. Wang, P. Chakraborty, T. Huang, J.-N. Hwang, and S. Lyu, "The 2019 ai city challenge," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[92] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International Conference on Machine Learning*, 2015, pp. 843–852.

[93] J. Harvey, Adam. LaPlace. (2019) Megapixels: Origins, ethics, and privacy implications of publicly available face recognition image datasets. [Online]. Available: https://megapixels.cc/duke_mtmc/

[94] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.