Video-based Person Re-Identification as a Markov Decision Process

Arulkumar S (CS15D202)

Indian Institute of Technology Madras

1 Introduction

Person Re-identification is the task of matching a person across multiple non-overlapping camera views. It is a common and fundamental task in Surveillance scenarios, Human-Computer interaction, Human activity analysis etc. Due to the recent surge in the usage of surveillance cameras, the need for automatic detection, tracking and re-identification of humans/objects is ever increasing to overcome the manual analysis of recorded videos. In this regard, there are recent attempts to solve automatic person re-identification, especially due to the emergence of Deep learning. Person re-identification can be based on an Image, where each person has only one image or, can be based on multiple frames per person. Having a set of frames per person allows the algorithms to acquire several clues (from different viewpoints, for example).

2 Video based Person Re-ID as an RL problem

There are several machine learning solutions proposed for video based person reid. Typically, the ReID methods first extract frame-level features from the images using either conventional (SIFT, HoG, LBP etc) or using Deep Neural Networks (DNNs). Then the frame level features are combined together by using some kind of temporal accumulation of features: max or average pooling of frame level descriptors[2], using RNN [4, 10, 1] to summarize the per-frame features etc., and the accumulated video level feature is used to classify if the frames belong to particular query person. Apart from these typical solutions, a recent work[9] has formulated this task in Reinforcement Learning (RL) setting (using Q-learning). Inspired by this work, in this project, we would like to re-implement the solution presented in [9] and analyze their performance.

The MDP formulation of Video based Re-ID method [9] in the literature is illustrated in the figure 1.

2.1 MDP formulation

In the MDP shown in figure 1, the Environment contains the video frames of two persons to be matched. The formulated MDP from [9] is a Discounted Finite Horizon MDP with t_{max} time stages, where t_{max} is a hyper-parameter denoting maximum possible total number of frames compared between two persons (Though there can be variable number of frames available for each person, the MDP formulation in [9] assumes that the frame lengths are equal for all persons). At every stage of the MDP, the environment provides the agent with a feature vector (state) extracted using machine learning techniques such as DNNs. The Agent can take one of the decisions of $\{$*same, different, unsure*$\}$ at each state. The actions *same* and *different* leads to a Terminal state T with positive and negative reward respectively. The action *unsure* will either make the environment to provide another feature vector to the agent or a negative reward if all the images are depleted. The objective of the agent is to optimally make decisions to maximize the reward.

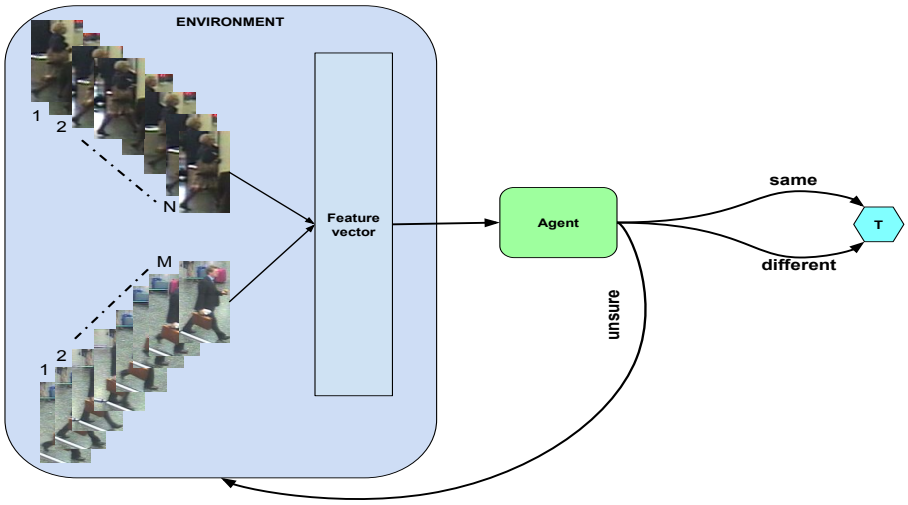


Figure 1: MDP formulation for the task of video based person re-identification. Here, *same, different, unsure* are actions.

Given the video frames of two persons and the extracted frame-level features (of d-dimension) from the frames, in mathematical notations, the Markov decision process $\{\text{S, A, T, R}\}$ is defined as follows:

States S_t: Each state consists of three components:

1. The absolute difference between the t^{th} frame descriptors of the persons to be compared (d-dimension)
2. Historical features up-to time step t to keep track of the accumulated features so far based on the previous decisions (d-dimension)
3. Hand-crafted features with mean, min, max norms of absolute difference features up-to time step t (3 dimension)

i.e., each state S_t is represented by a $2d + 3$ dimensional vector (refer figure 3).

Actions A_t & Transitions: At each time step t, given the state S_t, there are 3 actions possible:

- *same* - terminates immediately
- *different* - terminates immediately
- *unsure* - the feature corresponding to next image pair is given to the agent

Rewards R_t: The reward is defined as follows:

$$R_t = \begin{cases} +1, & \text{if } A_t \text{ matches ground truth (}\textit{same / different}\text{)}, \\ -1, & \text{if } A_t \text{ doesn't match the ground truth (}\textit{same / different}\text{) or } (t = t_{max} \ \& \ A_t \text{ is still } \textit{unsure}), \\ r_p, & t < t_{max} \text{ and } A_t \text{ is } \textit{unsure} \end{cases} \tag{1}$$

3 Model Architecture

3.1 Alexnet Base Network

As mentioned in [9], Alexnet [3] network pretrained with ImageNet dataset is chosen as the base (or feature extraction) network. From the base network, the last two fully connected layers (classification layer elements) are truncated. i.e., only convolutional feature extraction layers are retained. The retained model is appended with a frame level global-average-pooling[8] which takes a spatial average of every channel to obtain frame-level features, followed by a temporal average pooling layer to acquire a video-level descriptor. These video descriptors are passed through a Softmax layer to classify the video belonging to a particular person in training set. The video-level features are also used for metric-learning with Triplet loss.

The network architecture is shown in figure 2. The network takes N image frames of size $3(RGB) \times 224(Height) \times 112(Width)$ as input. The image frames are first passed through a Conv$(64$ channels$, 11 \times 11, stride = 4, pad = 2)$ + ReLU + Maxpool block and results in $64 \times 13 \times 5$ sized feature maps. Further, these feature maps are passed through 3 consecutive Conv$(3 \times 3, stride = 1, pad = 1)$ + ReLU blocks followed by one Conv$(3 \times 3, stride = 1, pad = 1)$ + ReLU + Maxpool block which gives out $192, 384, 256, 256$ channels respectively. At the end of the feature extraction pipeline, for every frame, the resulting feature maps are of size $256 \times 6 \times 2$. Further, each of the frame-level feature maps is passed through a global-average-pooling layer to get 256-dimensional descriptor for each frame. Once the frame-level descriptors are obtained, the video-level descriptor is acquired by averaging all the descriptors corresponding to all the frames' descriptors in the video.

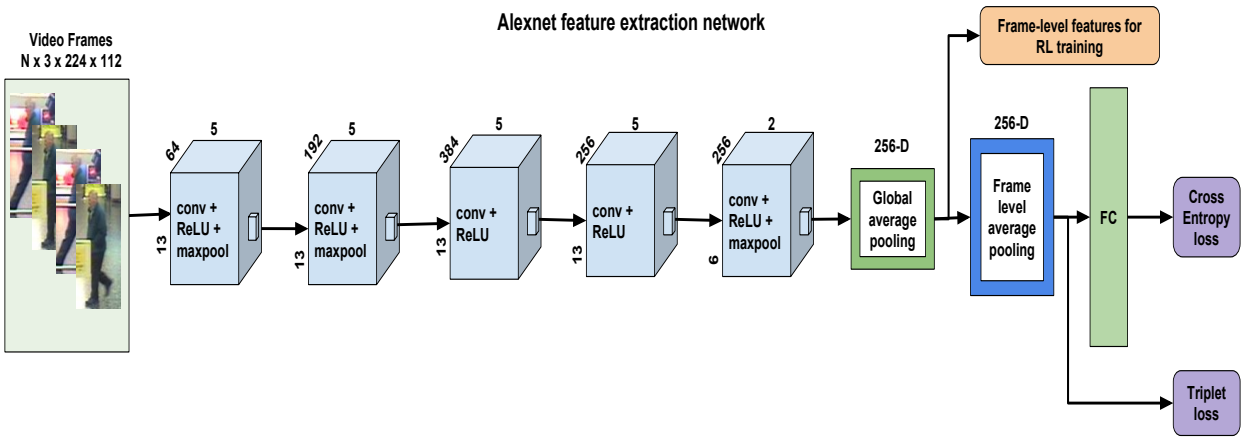


Figure 2: Alexnet base network used for pretraining and feature extraction to training RL-based DQN

3.2 Pretraining of the Base Network

As a first step, the base network (already pretrained in ImageNet dataset) is pretrained using the video based ReID dataset with the supervised training strategy consisting of Softmax Cross-Entropy loss and Triplet loss as shown in figure 2. It is inline with the method followed by [9].

3.3 Pretraining Losses

Let $\{X_i, y_i\}$ be the i^{th} $\{video, label\}$ pair in the training set, where y_i is corresponding to the unique person identity among P identities in the training set. The cross-entropy loss is defined as:

$$L_{softmax} = -\frac{1}{P}\sum_{i}\sum t_j^i \log p_j^i \tag{2}$$

where p_j^i is the softmax probability estimated by the base network for i^{th} example and j^{th} person. $t_j^i = 1$, if $y_i = j$.

For each sample a (designated as *anchor*) in the batch, the hardest positive and the hardest negative samples within the batch are selected when forming the triplets for computing the loss $L_{triplet}$:

$$L_{triplet} = \sum_{i=1}^{P}\sum_{a=1}^{K} \max\left(m + \max_{p=1,\ldots,K} D(f_a^i, f_p^i) - \min_{n=1,\ldots,K} D(f_a^i, f_n^i), 0\right) \tag{3}$$

Here, $D(.,.)$ denotes the distance between descriptors (in this case, L2 distance)m m is the margin, f_a, f_p, f_n signifies the descriptors of anchor, positive, negative samples respectively. Intuitively, the triplet loss makes the descriptors of the same persons (intra class) to be closer and the descriptors of different persons (inter class) to be farther.

Having defined both the supervised loss functions, the total loss is given by:

$$L = L_{softmax} + L_{triplet} \tag{4}$$

3.4 Deep Q-Learning Network (DQN) Architecture

In the second phase of training, the deep reinforcement Q-learning model is trained to make informed sequential decision if the given two video frames belong to same person or not. The Q-network consists of 2 fully-connected layers of 128 output nodes with ReLU activation followed by a fully connected layer of 3 output nodes. The 3 output nodes are used to predict the Q-values of 3-actions for every state (as explained in section 2.1).

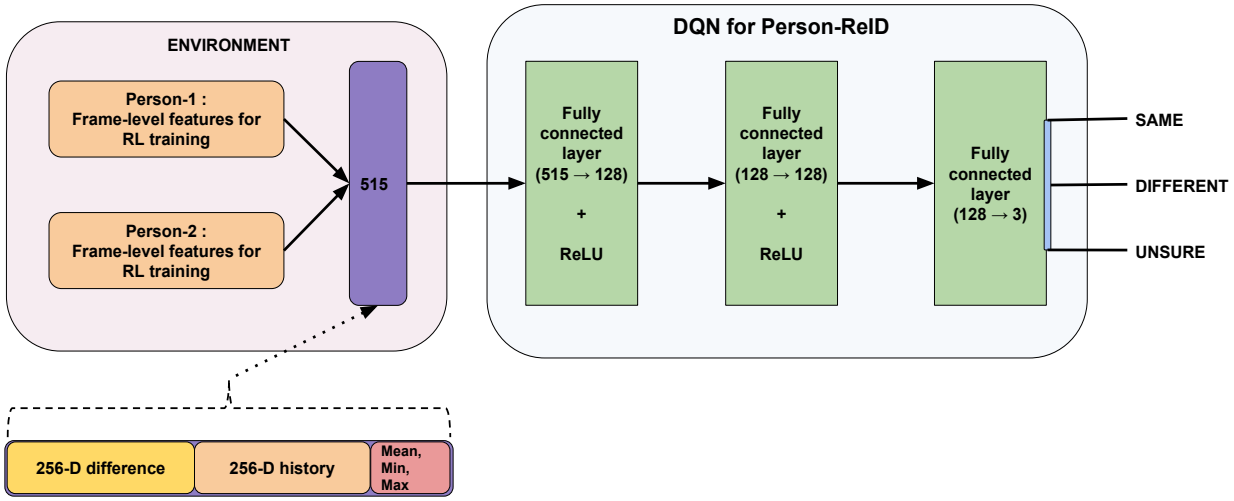


Figure 3: DQN architecture used for Person ReID Q-learning

The state vector at time t has 515 dimensions when using alexnet variant of base network (section 3.1). The Q-network takes as input a 515-dimensional state vector and passes it through 3 consecutive fully-connected layers to predict the state action values ($Q(S_t, A_t)$) corresponding to 3 actions.

History features: History features at t^{th} timestep of the episode is calculated as:

$$h_t = \frac{\sum_{i=1}^{t-1} w_i \times o_i}{\sum_{i=1}^{t-1} w_i} \tag{5}$$

here w_i is the weightage given to each of the states/features up to timestep t. w_i is estimated based on the significance of the features upto timestep t in making the correct decision and it is estimated as:

$$w_i = 1 - \frac{e^{Q_u}}{e^{Q_s} + e^{Q_d} + e^{Q_u}} \tag{6}$$

Here, $Q_s = Q(s_t, a_t = same), Q_d = Q(s_t, a_t = different), Q_u = Q(s_t, a_t = unsure)$. Intuitively, the frame features that the Q-network decides as *unsure* will have less weightage while keeping track of historical features.

3.5 DQN learning

The DQN learning is proceeded as follows: Similar to [9, 5], there are two important concepts used in DQN training to make the training converge.

Replay buffer: The agent consists of an experience replay memory. At each episode, the agent receives the state vector s_t from the environment and predicts an action for the particular state based on the predicted state-action values $Q(s_t, a_t)$. According to the action taken (a_t), the environment either provides the agent with the next state (s_{t+1}) or terminates the episode and the reward (r_t). The samples from the episode as a tuple form (s_t, a_t, r_t, s_{t+1}) are stored in replay buffer.

Target network: Once the data samples are collected and stored into replay buffer, the Q-network has to be trained to learn to approximate state-action values. In this regard, for each M iterations of training, the Q-network is backed up as Q_m and considered as the target network. At each training iteration, a batch of samples of the form (s_t, a_t, r_t, s_{t+1}) are sampled from replay buffer and the network is optimised by:

$$Q(s_t, a_t) = Q(s_t, a_t) + \eta(r_t + max_{a'_{t+1}} Q_m(s_{t+1}, a'_{t+1}) - Q(s_t, a_t)) \tag{7}$$

4 Experiments

4.1 Dataset

PRID 2011 : PRID2011 [10] is a medium-sized Video based ReID dataset with total 178 identities observed in 2 cameras. According to the defined protocol, the identities are randomly split into half for training and testing. (i.e, 89 identities for training and 89 identities for testing). Each image sequence has variable length consisting of 5 to 675 image frames, with an average number of 100 images. Images from both cameras contain variations in view-point, illumination, background and camera characteristics.

4.2 Implementation Details

The model architectures are implemented using PyTorch [6] framework. The code base is available in github repository: `https://github.com/InnovArul/personreid_sequential_rl`. The base network (Alexnet[3]) is loaded from `torchvision.models` package, which is pretrained using ImageNet dataset.

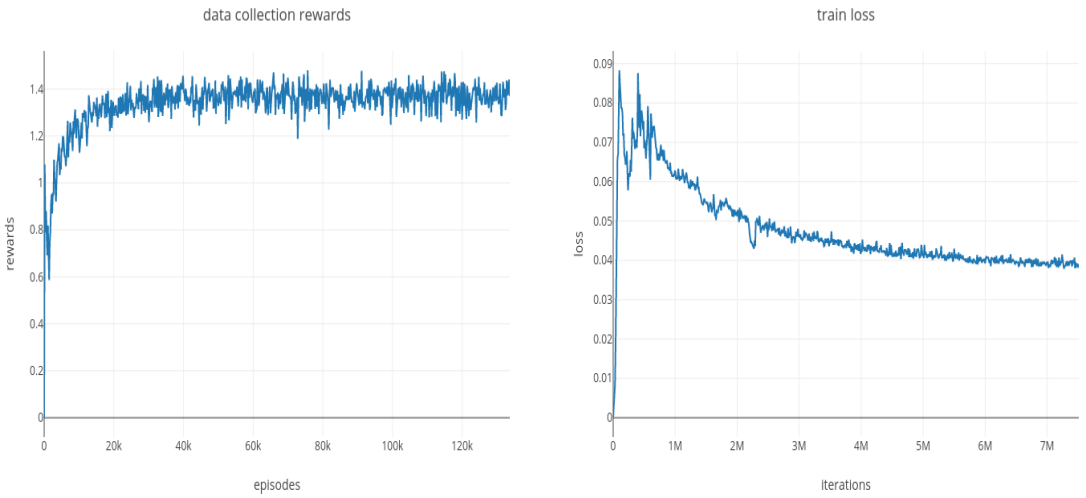
4.3 Training

During the supervised pretraining phase, the video frames are normalized using the mean and standard deviation from ImageNet dataset, random translation and horizontal flip are used for data augmentation. The hyperparameters used during supervised pretraining phase are: learning rate = 0.0003, weight decay = $5e^{-4}$, batch size = 32, margin $m = 0.9$. Adam optimizer is used for training.

For training with respect to Q-learning, the frame-level features are extracted using the pretrained base network. The base network weights are freezed while collecting the samples. During data collection, every identity is selected with a positive pair and a negative pair of video frames from the dataset. i.e, In PRID, there are 89 identities. Hence, 178 episodes of data collection at each epoch. After data collection, the training routine is carried out with following hyperparameters: Batch size of samples from replay buffer = 64, replay buffer memory size = 10000, the number of iterations per epoch = 10000, learning rate = 0.001, r_p = 0.2, weight decay = $5e^{-4}$, $t_{max} = 8$, discount factor = 0.99. Epsilon greedy exploration strategy is used while collecting data samples and Epsilon (ϵ) is varied according to the formula:

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \exp^{-\lambda N_{episode}} \tag{8}$$

Here $\epsilon_{min} = 0.1, \epsilon_{max} = 1, \lambda = 0.0001, N_{episode}$ = index of current episode.



(a) Average rewards collected during data collection ($r_p = 0.2, t_{max} = 8$)

(b) Training error while training DQN ($r_p = 0.2, t_{max} = 8$)

Time taken: For the baseline network training, it takes ~ 8 hours. For training Q-learning based DQN, it takes ~ 60 hours. The training is carried out in Nvidia GTX GeForce 1080Ti GPU. During testing time, each episode takes around 2.5 milliseconds.

4.4 Evaluation protocol

Person ReID is a retrieval based task which is evaluated based on ranking criteria. The evaluation method is as follows: For every query identity, the query feature will be extracted from his/her video frames. The extracted query features will be compared against a set of predefined gallery identity features to determine the distance between the features. The distances are sorted based on the gallery identity. A rank-N match is counted if the query identity is matched within top-N sorted identites from Gallery.

To evaluate the baseline method, the descriptors of query and gallery are compared and the ranking is carried out using L2 distance. For evaluating the performance of RL based DQN, the score is considered as the difference between the state-action values ($Q(s_N, same) - Q(s_N, different)$) at the end of the episode. Here, s_N denotes the final state of the episode.

4.5 Quantitative evaluation

Method	mAP (%)	rank-1 (%)	rank-5 (%)	rank-10 (%)	rank-20 (%)	avg. #frames
RNN-CNN [4]	-	70.00	90.00	95.00	97.00	100
Two stream [1]	-	78.00	94.00	97.00	99.00	100
CNN +XQDA [10]	-	77.9	93.5	-	99.3	100
baseline (alexnet)	86.2	80.9	93.3	96.6	100.0	100
Q-learning ($r_p = 0.2$)	81.9	76.4	88.8	95.5	97.8	**3.949**
Q-learning ($r_p = 0.3$)	75.7	66.3	87.8	91.0	94.4	**5.05**

Table 1: Performance of the Q-learning based Person ReID method in PRID2011 dataset compared to other state-of-the-art methods in the literature (**mAP** denotes Mean Average Precision (higher the value, better the method is), **rank-N** denotes ranked accuracy (higher the value, better the method is))

As shown in the table 1, the experimented method based on Deep Q-learning relatively performs on-par with the methods in the literature while using less number of frame comparisons.

4.6 Qualitative results

Figure 5: Query (left), Gallery (right). Taken actions are *unsure, unsure, unsure, different*

4.7 Discussion & Status

- The RL based Q-learning method is intuitive and interpretable. By looking into the actual images during test, it is evident that the method chooses to take action *unsure* for the images that are having

Figure 6: Query (left), Gallery (right). Taken actions are *unsure, different*

Figure 7: Query (left), Gallery (right). Taken actions are *unsure, different*

Figure 8: Query (left), Gallery (right). Taken actions are *unsure, same*

more similarity and hard to predict. When the method is confident enough and the images are clear, the action *same, different* are taken.

- A downside of RL based approaches compared to other CNN based approaches with respect to Person ReID is that since the task is used in video surveillance and retrieval, the need for real time performance is expected. We noticed that even though the RL based approach is interpretable and intuitive and although only less number of images are compared, the inference time for large datasets (e.g., MARS) is pretty high than we can afford. This is due to the sequential nature of episodes. When we naively programmed, our code is very slow. Hence, there is a need for parallelized execution of episodes. For example, when working with large datasets during initial run, it took $\sim$ 15 hours just to evaluate the algorithm on its test set. The MARS[7] dataset has 1980 query videos and 9330 gallery videos to be matches. Hence, close to $9330 \times 1980 =\sim 18.5M$ episodes to be run. When each episode takes ~ 2.5 milliseconds, it takes around ~ 14 hours just to evaluate, whereas the baseline work which gives comparative performance takes only ~ 1 hour due to parallelization using GPUs. The RL can be sped up by batch processing of the video-frames, which we have not done for this

project.

- The adaptation of the method using Policy gradient algorithm is implemented and the code is available in the github repository. The policy did not converge and the performance is sub-optimal so far.

5 Conclusion

In this project, we have shown a practical application of Q-learning in the task of Person Re-identification[9]. We implemented the project from scratch and verified that the Q-learning based solution performs on-par with other state-of-the-art solutions available in the literature. As shown in the experiments, the RL based sequential decision process accounts for interpretability and intuitiveness.

References

[1] Dahjung Chung, Khalid Tahboub, and Edward J Delp. A two stream siamese convolutional neural network for person re-identification. In *The IEEE international conference on computer vision (ICCV)*, 2017.

[2] Jiyang Gao and Ram Nevatia. Revisiting temporal modeling for video-based person reid. *arXiv preprint arXiv:1805.02104*, 2018.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Niall McLaughlin, Jesus Martinez del Rincon, and Paul Miller. Recurrent convolutional network for video-based person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1325–1334, 2016.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[7] *MARS: A Video Benchmark for Large-Scale Person Re-identification*, 2016. Springer.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[9] Jianfu Zhang, Naiyan Wang, and Liqing Zhang. Multi-shot pedestrian re-identification via sequential decision making. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1116–1124, 2015.

Appendix

5.1 Coding details

The data loading part of the code is adapted from the github repository: `https://github.com/jiyanggao/Video-Person-ReID`.

The model is coded following the style of `gym` open source package, which provides virtual environments like *CartPole, MountainCar, Humanoids* etc., to experiment several RL algorithms.

The classes and their descriptions are given below:

5.1.1 `Environment`

File: `src/models/RL_model.py`

Like `gym` environments, this class provides member methods such as `reset(), step()` to interact with the environment.

5.1.2 `Memory`

File: `src/models/RL_model.py`

Implements the replay buffer. Provides APIs to insert the samples during the data collection phase and to sample a batch of samples during training.

5.1.3 `Brain`

File: `src/models/RL_model.py`

Implements the model used for Q-learning. It has APIs to `predict` Q-values for the given state.

5.1.4 `Agent`

File: `src/models/RL_model.py`

Encloses an instance of `Memory` and `Brain`. Given an `Environment`, the `Agent` deals with collecting data and training the `Brain` to predict correct actions according to the state.