Multimedia Tools and Applications (2024) 83:73569–73582
https://doi.org/10.1007/s11042-024-19111-0

1232: HUMAN-CENTRIC MULTIMEDIA ANALYSIS

# Person re-identification on lightweight devices: end-to-end approach

**Tuan Linh Dang[1] · Trung Hieu Pham[1] · Duc Loc Le[1] · Xuan Tung Tran[1] · Hoang Nam Le[1] · Khanh Hung Nguyen[1] · Tran Tuan Nghia Trinh[1]**

Received: 14 May 2023 / Revised: 23 January 2024 / Accepted: 27 March 2024 /
Published online: 12 April 2024


**Abstract**
In this work, an efficient and real-time person re-identification system based on an affordable hybrid framework was presented. The proposed pipeline consisting of human detecting, tracking and extracting features was developed based on lightweight deep neural models so that they could be computationally accelerated on limited hardware resources devices. A comprehensive and substantial dataset has been established aiming to facilitate the training and evaluation of a surveillance system implemented to monitor individuals in an indoor environment. The proposed processing pipeline was implemented on both low-cost devices as Nvidia Jetson Nano and Google Coral. The experimental results indicated that the system could achieve real-time performance with up to 29 FPS and 0.96 mAP for the person detection algorithm task via edge devices, whereas a comparable accuracy was reached on the proposed feature extraction model with 0.85 mAP.

**Keywords** Person re-identification · Lightweight architecture · Detection · Feature extraction

✉ Tuan Linh Dang
linhdt@soict.hust.edu.vn

Trung Hieu Pham
hieu.pt194763@sis.hust.edu.vn

Duc Loc Le
loc.ld211033M@sis.hust.edu.vn

Xuan Tung Tran
tung.tx194405@sis.hust.edu.vn

Hoang Nam Le
nam.lh207621@sis.hust.edu.vn

Khanh Hung Nguyen
hung.nk194297@sis.hust.edu.vn

Tran Tuan Nghia Trinh
nghia.ttt207623@sis.hust.edu.vn

[1] School of Information and Communications Technology, Hanoi University of Science and Technology, 01 Dai Co Viet road, Hanoi 100000, Vietnam

## 1 Introduction

Smart surveillance system which provides automatic analysis of streaming videos without human interference is getting more and more demanding in recent years [1]. In a traditional system, raw video data recorded by cameras is transmitted to a security center and manually monitored by officers. This method is not only prone to human errors but also demands tremendous human and financial resources. With the development in the technology of both hardware and software, an end-to-end camera system now can self-analyze information from video data and raise advanced actions automatically such as security warnings, detecting anomalous behaviors, and attendance tracking.

Person re-identification (person re-ID) is a novel research field in automated monitoring systems. The operation of acknowledgment of an individual from different video frames captured by multiple cameras is the objective of the person re-ID algorithm. This technique can be necessary for many computer vision-based applications, especially those that involve tracking entities across a system composed of multiple cameras. Most existing end-to-end person re-ID systems are deployed via two main approaches: cloud-based and edge-based [2]. The former mainly relies on a cloud server resource capable of applying complex algorithms, resulting in better recognition accuracy. Transmitting a large amount of raw video data from cameras to the cloud, however, causes extra workload on the network's bandwidth. This usually results in the degradation of the system's real-time performance. The latter approach, that being edge-based deployment, requires an expensive embedded device that is deployed on-site, in order to implement complex deep learning models (models such as the Jetson Nano AGX or the Jetson TX2) [2–4]. This leads to challenges in scaling smart surveillance systems across a broad area for low-budget users, such as shophouses, restaurants, and small buildings. Given the aforementioned advantages and disadvantages of cloud-based and edge-based deployment, this research paper henceforth proposes a hybrid edge-cloud person re-ID framework based on two low-cost embedded platforms, Jetson Nano and Google Coral [5].

Notable achievements have been recently made on person re-ID methods for edge devices. For instance, a mixed precision approach using lightweight models (models such as ResNet50 and MobileNet-v2) was implemented on an NVIDIA AGX Xavier embedded system, in order to form an end-to-end re-ID approach [6]. Another example is a combined framework of embedded devices, fog processing, and a cloud server based on a segmented binarized ResNet model, which was proposed in the previous paper [7]. Lastly, the "deep squared similarity learning method", as proposed in [8] involved learning similarities between different pairs of human images. That being said, the issues regarding pragmatic edge-based implementation could not be solved by the above-mentioned works.

Our primary research contribution involves developing a highly scalable, lightweight end-to-end re-identification framework. Specifically designed for deployment on resource-constrained devices, such as the Jetson Nano, our framework emphasizes efficiency without compromising performance. We introduced a tailored lightweight model, showcasing commendable results with significantly reduced latency. Additionally, we proposed our dataset, the BKREID dataset, which comprises 23,160 indoor images capturing 37 individuals.

The structure of our manuscript consists of five sections. Section 2 gives the information on related works. Our approach is presented in Section 3. Detailed information about the experiments can be found in Section 4. The final section, Section 5, summarizes this manuscript.

## 2 Related work

### 2.1 Re-identification systems

A person re-identification system is a model used to identify people if they leave a camera's line-of-sight, and then re-appear in another camera's line-of-sight. It is also used for setups that use multiple cameras at the same time to track specific entities. While being constantly utilized in security, traffic, or commercial centers, a person re-ID system faces many challenges, such as differences in lighting, viewing angles, environments, outfits worn by entities, etc. There is also the complexity involved in deploying the person re-ID system on multiple devices, as well as the cost of deploying the said system.

The End-to-end person re-identification [2] provides a method of re-identifying people on the video surveillance system that is actually deployed on the cloud system and edge devices, but the model is deployed mostly on devices. To achieve optimal performance, addressing such marginalization necessitates a robust system, which can result in high costs and challenges when attempting to deploy it across various devices and locations.

To solve this problem, we have proposed cheaper, more suitable models that can be mounted more easily on edge devices. These models will have the ability to identify people more quickly and efficiently in real-time.

### 2.2 Background

#### 2.2.1 Object detection

This computer vision task finds applications in various fields like transportation, security, surveillance, and robotics. In recent years, object detection tasks have experienced significant performance improvements, primarily by the appearance of deep learning. In this situation, Regions with CNN Features (R-CNN) [9], were among the pioneer studies. This approach used a CNN along with a region proposal algorithm. By combining these components, the system could accurately identify and categorize objects within the image data. Later, the authors extended R-CNN to Fast R-CNN [10], and Faster R-CNN [11], which improved the speed and accuracy of the original method by sharing computation across region proposals, and by introducing a region proposal network. Though achieving great accuracy, this type of two-staged approach is often considered slow and hard to implement in real-time. On the other hand, You Only Look Once (YOLO) [12] detects objects in a single forward pass through a single network. In this situation, images are partitioned into a grid of cells, from which the model then predicts objectness scores, class probabilities, and bounding box coordinates, for each cell. While fast and efficient, there are possibilities where said method struggles with small objects and densely-packed scenes. In addition, there are also many other approaches and variations that have been suggested for object detection, such as RetinaNet [13], which introduces a focal loss function that down-weights the contribution of easy examples and focuses on hard examples during training. CenterNet [14] simplifies the object detection pipeline by predicting their center points and sizes, rather than using anchor boxes or region proposals. Transformer-based methods were also introduced, such as DETR [15], in which a CNN backbone is leveraged to get features from an image. Multiple transformer blocks are then applied to the sequence of features to get a richer representation, before finally outputting a set of object classes and bounding boxes.

### 2.2.2 Object tracking

While object detection only detects objects in independent frames without providing any information about the objects' trajectories or identities over time, object tracking aims to follow an object's movements and maintain its identity, even when the object undergoes changes in appearance or motion. Various methods for object tracking have been developed; a traditional example would be Multiple Hypothesis Tracking (MHT) [16], which creates multiple tracks for each object, and uses a data association algorithm to select the most likely track at each time step.

Recently, great successes in deep learning and object detection have enabled tracking-by-detection methods to gain more and more popularity. As an example, SORT (Simple Online and Realtime Tracking) [17] provides a lightweight, simple object tracking algorithm that combines object detection and motion models. This approach has a Kalman filter to infer the state of each object in each frame based on the previous state, and the Hungarian algorithm is then used to associate the predicted states with the detected objects of the current frame. DeepSORT [18] extends on SORT by introducing a Siamese network that is pre-trained to differentiate between people, adding visual information to the process of association, and showing significant improvement in scenarios where objects are moving at high speeds. Other appearance-based methods were also widely used, such as a correlation filter [19, 20], which trains a filter that maximizes the correlation between the object template and the image patches. Another method is the color histogram-based tracker [21, 22], which uses color histograms to represent the appearance of objects and to track their movements.

### 2.2.3 Deep metric learning

This technique trains a neural network to learn a distance metric between data points. The objective is to discover a data representation so that related data points are mapped closer to the learned space. This objective is achieved through the optimization of a loss function, which encourages the network to minimize the gap between similar samples while maximizing the gap between dissimilar samples. The learned representation can subsequently be utilized for other tasks like clustering [23] or classification [24] tasks. Deep metric learning has shown promising results in many real-life applications like image retrieval [25] and face recognition.

### 2.2.4 Convert models

**a) TensorRT conversion**

NVIDIA has created TensorRT, an SDK for deep learning inference, which has the capability to import trained models from state-of-the-art deep learning frameworks [26]. It may also generate efficient runtime engines that can be easily deployed in embedded systems

TensorRT performs optimization of the neural network graph through various transformations that enhance throughput and minimize latency during the inference process. These transformations within the graph improve operation speed and efficiency without altering the graphs' underlying computation. The process of INT8 and FP16 quantization technique and optimized runtime engine generation in TensorRT significantly accelerate the inference speed of neural networks up to 8 times.

For embedded cameras with constrained resources, utilizing TensorRT helps in the processing of large amounts of video data efficiently and accurately, while keeping latency and memory usage low throughout.

**b) Tflite conversion**

TensorFlow Lite (TFLite) is a lightweight framework for building machine learning applications that can run on mobile and embedded devices. TFLite is designed to be fast, efficient, and easy to use, making it an ideal choice for developers who want to deploy machine learning models on edge devices [27]. One significant advantage of using TFLite on edge devices is the ability to leverage the power of edge TPUs (Tensor Processing Units). Edge TPUs are custom-designed chips optimized for running machine learning workloads on edge devices. By using TFLite on edge TPUs, developers can achieve significant performance improvements, lower latency, and lower power consumption, making it possible to build real-time, low-power applications including image recognition or object detection. Overall, TFLite provides a powerful and flexible framework for building machine learning applications that can run on the edge, with the added advantage of being able to leverage the power of edge TPUs.

## 3 Proposed framework

### 3.1 Overview

This research presents a comprehensive system capable of monitoring entities that enter or exit a building or room as demonstrated in Fig. 1. Our suggested system consists of three components: two edge devices and one server. One of the edge devices will be positioned at the entrance of the building or room; the other will be positioned at the exit. Said edge devices will detect humans and provide the bounding box and frame information to the server. Next, the server runs tracking and feature extraction algorithms. Each individual entity's extracted features (as detected by the entrance device) will then be saved in a database. The features of individuals captured by the exit device will be matched with those present in the database by estimating the distance between each pair of feature vectors. Hence, we will be able to re-identify these individuals as long as they have already been spotted using the entrance device.

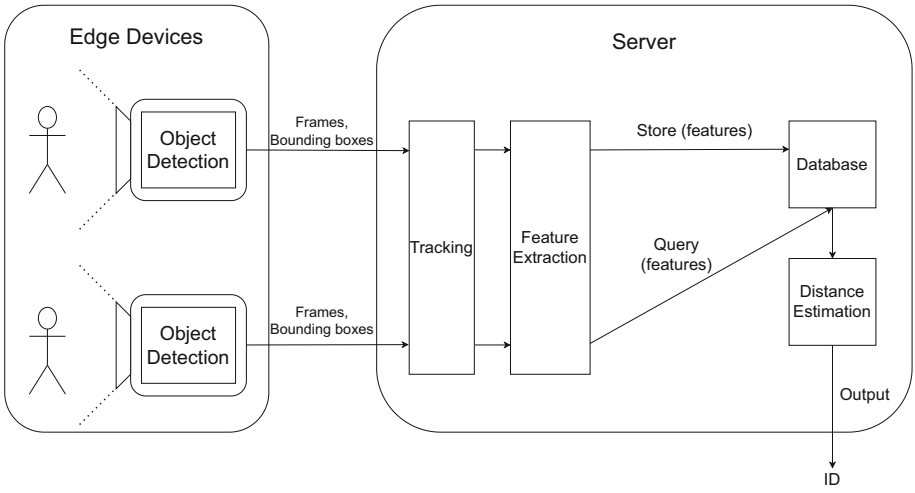


**Fig. 1** The proposed re-identification system has an Edge Devices block for object detection and a Server block for person tracking and feature extraction

## 3.2 Modules on edge devices

### 3.2.1 Object detection

As discussed above, YOLO is a one-stage object detector well-known for its efficiency and accuracy. For the purpose of our experiment, with our detection module's parameters to be accurate while also as fast as possible, so we chose the stripped-down version of YOLOv5 being YOLOv5n. YOLOv5n is designed to be lightweight by incorporating fewer convolutional layers and anchor boxes compared to the base version. This reduction in components leads to a decrease in parameters and FLOPS count, thus more efficient. This is particularly important as our detection module is intended for use on edge devices, where computational resources are limited. We also tested the model against several bigger versions and observed no significant differences in terms of performance on our datasets, so we chose YOLOv5n due to its ease of deployment and high efficiency. Our research took advantage by using the pre-trained weights of a previous study without the need for fine-tuning, as this model had already been trained on COCO dataset and was able to recognize the "person" class [28], which is the primary object of interest in our device's application.

Despite the model's lightweight characteristics, our testing showed that it was unable to achieve real-time performance on devices with limited resources when used with raw PyTorch. This was due to a combination of factors, including insufficient computing power and a computing graph that was not optimized for the hardware. Thus, it is necessary to transform the model into an optimized format tailored to each device, specifically into the TensorRT format for Jetson Nano devices, and the TFLite format for Raspberry Pi devices (along with Google Coral). YOLOv5n could be easily converted to the ONNX format, and then to the TensorRT format. Parameters of the model are quantized down to 16-bit floating point, which is a good choice for the speed-accuracy trade-off. In the case of the TFLite format conversion, INT8 weights were used as the TPU only supports INT8 inference.

## 3.3 Modules on server

### 3.3.1 Person tracking

After detection, objects must be tracked in order to maintain their identities across frames. For this part, we researched several methods and chose the SORT algorithm because of its efficiency and simplicity. This algorithm can track multiple objects simultaneously and assign them unique IDs, even when objects are close to each other, or when occlusions and temporary disappearances occur. This makes the SORT algorithm highly suitable for tracking in crowded environments, with many individuals in close proximity. In addition, the algorithm is very fast, owing to it using only simple math calculations. This is particularly important for our purposes, as our tracking module is placed on the server; the server needs to be able to handle tracking simultaneously for multiple camera streams from edge devices

### 3.3.2 Feature extraction

The obtained bounding box information from previous parts will be used to crop regions of interest from images. After that, the features of each region will be extracted using a deep learning model. We proposed a lightweight, customized model that obtains good results in relative to a reduced latency.

**Backbone** We proposed a customized backbone based on MobileNetV2 [29] due to its low computational cost and lightweight characteristics. Before feeding the input into the model, we first resize it to $128 \times 64$. Because of the small resolutions, only a portion of MobileNetV2 is required. Our backbone's output is the expanding layer of the $12^{th}$ block in the original model. A global max pooling layer followed by a 224-dimensional fully connected layer are added to the backbone. In addition, a dropout layer is added after the global max pooling layer to address any overfitting issues.

**Loss function** For our objective loss function, we utilized the Hard Triplet Loss [30]. In a batch, the loss motivates the maximum distance between a pair of embeddings from the same class to be less than the minimum distance between a negative pair plus the margin constant. When constructing the triplets for calculating the loss, the hardest negative and positive samples in the batch were selected.

#### 3.3.3 Distance estimation

After extracting the features of individuals from exit devices, we will then compare them with features present in the database. A distance will be calculated between each pair of feature vectors to determine if they are from the same individual, using cosine similarity as our distance metric.

## 4 Experiment

### 4.1 Environmental setup

Our experiments employed the Google Colab cloud environment and a computer that had an AMD Ryzen 5 5600H CPU equipped with NVIDIA GeForce RTX 3050 Ti 4GB GPU for the training of our models. Each camera node in the system is either a budget-friendly NVIDIA Jetson Nano device or a Google Coral device, that is specifically made for individual detection. However, only simple, computer-based, lightweight devices were used for the final experimental results presented in this work. As for deep learning frameworks, we utilized TensorFlow and TensorRT (Table 1).

### 4.2 Hyper-parameters and training settings

**Object Detection:** Our experiment utilized the published weights of YOLOv5n trained on the COCO datasets, which already contain the class "person". Our network required the images to be resized to the dimension of $416 \times 416$ before they are processed.

**Table 1** Hardware setup on the server and at the edge

| | Jetson Nano | Raspberry Pi + Google Coral | Server |
|---|---|---|---|
| CPU | quad-core ARM®Cortex®-A57 | Broadcom BCM2711 | AMD Ryzen 5 5600H |
| GPU | 128-core Maxwell™ | None | NVIDIA GeForce RTX 3050 Ti 4GB |
| Memory | 4GB 64-bit LPDDR4 | 4GB LPDDR4 SDRAM | 8GB |
| TPU | None | Google Coral TPU | None |

**Feature extraction** The hyper-parameters for this model are summarized in Table 2 First, we resized the input to $128 \times 64$. Second, the resized tensor will be divided by 255 to normalize the input from zero to one. Then, we chose a batch size of 64. In a training step, we would choose a person in the dataset, and then take at most half of the batch size of the samples from that person. The remaining samples in the batch will be taken randomly from other people. An epoch would end when we iterate through all of the people in the training set. We choose the margin constant for the Hard Triplet Loss [30] to be 0.2. Afterwards, our model would be trained for 500 epochs, and the model that obtained the highest top-1 accuracy on the validation set was considered the final model.

## 4.3 Datasets

### 4.3.1 Penn-Fudan

Though we decided not to fine-tune our detection model, we still needed a dataset in order to benchmark our model's performance. For this purpose, we chose the public Penn-Fudan Dataset, published in accordance with [31]. The dataset contains images, bounding boxes, and segmentation masks of people taken around campuses and urban streets. The number of pedestrians per image ranges from 1 to a maximum of 8, with 170 images in total, 345 labeled persons, and 423 bounding boxes.

### 4.3.2 CUHK03

This dataset has 14,097 photographs of 1,467 individuals [32]. Six campus cameras were used for the image collection, with two cameras for each identity. In this dataset, there are two categories of bounding box annotations: manual labeling, and automatic labeling using a detector. This dataset also provides 20 train/test partitions, where 100 identities are reserved for testing, and the remaining samples are considered training samples. Our approach only investigates the manually-labeled part for training, evaluation, and comparison.

### 4.3.3 BKREID

Our proposed dataset's name is BKREID; it is used to develop person re-identification models, and to contribute to improving the system's efficiency. This dataset was taken indoors with stable light, so it will not be affected by weather factors and outside light, compared to the CUHK03 [32] dataset (that was taken outdoors).

The proposed dataset consists of four rounds of videos, each round consisting of two videos corresponding to two cameras placed at two locations: the entrance and the exit. The first camera captures the image of individuals entering at the entrance; this camera is used primarily to collect images. The second camera captures the image of individuals leaving at the exit; this camera is used primarily to determine their identities. The dataset contains 23,160 images of 37 people, and is carefully labeled with the aid of an object detector. The dataset is used for training and evaluation purposes (Fig. 2).

**Table 2** Hyperparameters for feature extraction model

| Input size | Batch size | Margin | Number of epochs |
|---|---|---|---|
| $128 \times 64$ | 64 | 0.2 | 500 |

## 4.4 Experimental results

### 4.4.1 Results for detection model

The performance of the detection module on the edge devices via two scenarios was investigated: first, running the module in raw PyTorch, and second, running the module after converting it to TensorRT/TFLite. The model processed all the images to produce the initial output and then applied non-maximum suppression (NMS) to eliminate overlapping boxes and generate the ultimate bounding box predictions. Metrics were then calculated after NMS. Both the NMS intersection-over-union ratio (IoU) and confidence threshold were set to 0.45 and 0.25, respectively, in both scenarios. That means only boxes with a confidence score equal to or greater than 0.25 were considered, and two boxes were called overlapping if their IoU was above or equal to 0.45.

Tables 3 presented the AP@0.5 and mAP metrics of three different lightweight detection models evaluated on two datasets and their processing time, measured in FPS. After converting the models to optimized formats, the results indicated that the models' inference speed


**Fig. 2** Images from BKREID Dataset

**Table 3** Results of Detection Module before and after conversion. The numbers before "/" are the results before conversion. The numbers after "/" are the results after conversion

| Model | Dataset | Jetson Nano | | | Raspberry Pi + Coral TPU | | |
|---|---|---|---|---|---|---|---|
| | | AP@0.5 | mAP | FPS | AP@0.5 | mAP | FPS |
| YOLOv5n | Penn-Fudan | 0.956/0.956 | 0.963/0.962 | 13/50 | 0.948/0.942 | 0.953/0.952 | 4/23 |
| | BKREID | 0.962/0.958 | 0.965/0.962 | 14/50 | 0.917/0.894 | 0.920/0.897 | 3/22 |
| YOLOv7-tiny | Penn-Fudan | 0.975/0.973 | 0.978/0.977 | 9/31 | - | - | - |
| | BKREID | 0.979/0.977 | 0.981/0.980 | 9/31 | - | - | - |
| YOLOv8n | Penn-Fudan | 0.967/0.967 | 0.972/0.971 | 11/40 | - | - | - |
| | BKREID | 0.975/0.972 | 0.977/0.975 | 11/40 | - | - | - |

increased by approximately 4-5 times on the Jetson Nano and Raspberry Pi with Google Coral TPU. The performance degradation was minimal. YOLOv5n on the Jetson Nano performed well at around 0.96 mAP while delivering real-time speed. As for the Raspberry Pi (with Google Coral), there was a marginal decline in the mAP due to the weight quantization from 32-bit floating-point to 8-bit integer. The impact, however, remained minimal. The results showed that converting a model can significantly increase speed without sacrificing accuracy.

We also see that the detection accuracy across three YOLO variants did not differ much and should not significantly affect the system result (some comparisons of the detected bounding boxes can be seen in Fig. 3). Tables 4 provided info about each model's memory cost and

**Fig. 3** Detected bounding boxes from three models. Green: YOLOv5, Red: YOLOv7, Blue: YOLOv8

**Table 4** Memory usage and computational complexity of different lightweight detection models

| Model | GPU Memory (MB) | #Params (M) | FLOPS (B) |
|---|---|---|---|
| YOLOv5n | 170 | 1.9 | 1.9 |
| YOLOv7-tiny | 189 | 6.2 | 5.8 |
| YOLOv8n | 172 | 3.2 | 3.7 |

computational complexity, and YOLOv5n still appeared most efficient. That is why we chose YOLOv5n to be our detection module.

### 4.4.2 Results for re-identification model

Table 5 shows the experimental results with publicly recognized CUHK03 and proposed BKREID datasets. In addition, two different versions of our model were investigated. The first model is a native Tensorflow model, and the second one is a TensorRT-converted model.

There was no difference in accuracy or parameter count between the two versions of our models. However, the converted model exhibited superior speed and memory utilization performance. In addition, our proposed model has shown results that are on par with state-of-the-art models in the CUHK03 dataset while also delivering superior speed and memory consumption efficiency compared to these models. Our model's exceptionally lightweight and fast characteristics made it highly suitable for our proposed system. As the number of cameras increased, this model did not significantly burden the system when executed multiple times. Even the author of the DiP model [34] does not disclose the number of parameters; this state-of-the-art model employs a vision transformer as its backbone, with the base version having 86 million parameters. Thus, our model exhibited a weight reduction of over a hundredfold compared to this model. Moreover, in terms of frames per second (FPS), the performance of our converted model showcased a remarkable enhancement, exhibiting a speed that is more than 9 times faster post-conversion. This speed advantage extended to over 150 times compared to the lightMBN model and nearly 35 times compared to the FPB model. Additionally, our approach also produced decent results on the BKREID dataset.

### 4.4.3 Results for system

In our work, we have tested with two-person re-identification scenarios. The first scenario consisted of two Jetson Nano devices and a server, while the second scenario consisted of a Jetson Nano device, a Google Coral device, and a server. At the time of our testing, the results show that both test scenarios are good and stable. The system speed averages around

**Table 5** Results of feature extraction models on the CUHK03 and BKREID datasets before and after conversion

| Model | GPU Memory (MB) | #Params (M) | FPS | CUHK03 (mAP) | BKREID (mAP) |
|---|---|---|---|---|---|
| Ours (Before conversion) | 924 | 0.63 | 199 | 0.850 | 0.734 |
| Ours (After conversion) | 390 | 0.63 | 1810 | 0.850 | 0.734 |
| LightMBN[33] | 1120 | 9.15 | 12 | 0.851 | - |
| DiP [34] (ViT [35] backbone) | non-disclosed | non-disclosed | non-disclosed | 0.856 | - |
| FPB[36] | 2864 | 27.08 | 52 | 0.838 | - |

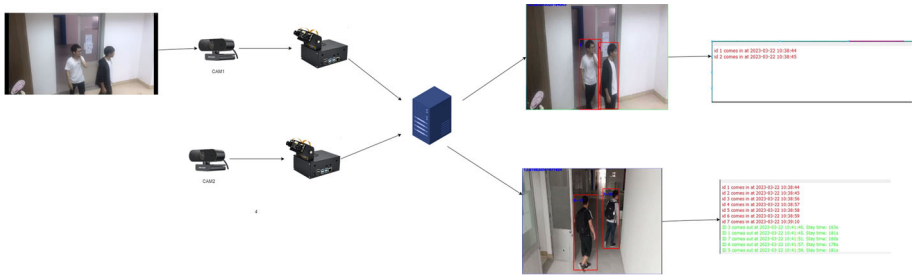

**Fig. 4** The operation of our system

13-15 FPS. The first scenario had a higher accuracy of person re-id than the second scenario. The discrepancy in accuracy can be attributed to the fact that the first scenario's model only required a conversion from FP32 to FP16 datatype, whereas the second system's model required a conversion from FP32 to INT8 datatype. The operation of our system may be observed in Fig. 4.

## 5 Conclusion

Our manuscript investigates and proposes an end-to-end person re-identification approach for resource-constrained devices. This approach incorporated object detection, tracking, and feature extraction. In addition, a novel architecture for the feature extraction model is also presented, as well as a new dataset, BKREID. Our proposed model achieved both fast inference speed and accurate results on evaluated datasets. TensorRT and TFLite were utilized to optimize our models, which can now operate efficiently on edge devices, such as Jetson Nanos. We tested our entire pipeline and its individual modules on numerous datasets and obtained satisfactory results.

A possible direction for future improvement is to use quantization and to efficiently optimize each model so that all of the models could be implemented on edge devices without the help of an additional server. Our system is scalable and can be investigated with various real-life applications, like supervision in schools, supermarkets, or stores.

**Acknowledgements** This research is funded by Hanoi University of Science and Technology (HUST) under project number T2022-PC-052.

**Data Availability** Accessing our dataset is available from the corresponding author upon reasonable request.

## Declarations

**Conflicts of interest** The authors declared that they have no conflicts of interest with regard to this work.

## References

1. Elharrouss O, Almaadeed N, Al-Maadeed S (2021) A review of video surveillance systems. J Vis Commun Image Represent 77:103116
2. Gaikwad B, Karmakar A (2022) End-to-end person re-identification: Real-time video surveillance over edge-cloud environment. Comput Electr Eng 99:107824

3. Neff C, Mendieta M, Mohan S, Baharani M, Rogers S, Tabkhi H (2019) Revamp 2 t: real-time edge video analytics for multicamera privacy-aware pedestrian tracking. IEEE Internet of Things J 7(4):2591–2602
4. Nvidia (2023) Jetson Accelerating Next-Gen Edge AI and Robotics. https://www.nvidia.com/en-in/autonomous-machines/embedded-systems/. Accessed: 18-April-2023
5. Google (2023) Coral USB Accelerator. https://coral.ai/products/accelerator/. Accessed: 18-April-2023
6. Baharani M, Mohan S, Tabkhi H (2019) Real-time person re-identification at the edge: A mixed precision approach. In: Image analysis and recognition: 16th international conference, ICIAR 2019, Waterloo, ON, Canada, August 27–29, 2019, Proceedings, Part II 16, pp. 27–39. Springer
7. Chen Y, Yang T, Li C, Zhang Y (2020) A binarized segmented resnet based on edge computing for re-identification. Sensors 20(23):6902
8. Chen X, Li Z, Xiao S, Chen Y (2018) Deep square similarity learning for person re-identification in the edge computing system. In: 2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData), pp 561–567. https://doi.org/10.1109/Cybermatics_2018.2018.00117
9. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28
10. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision (ICCV)
11. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28
12. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
13. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
14. Zhou X, Wang D, Krähenbühl P (2019) Objects as points. arXiv:1904.07850
15. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: Vedaldi A, Bischof H, Brox T, Frahm J-M (eds) Computer Vision - ECCV 2020. Springer, Cham, pp 213–229
16. Reid D (1979) An algorithm for tracking multiple targets. IEEE Trans Autom Control 24(6):843–854
17. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B (2016) Simple online and realtime tracking. In: 2016 IEEE international conference on image processing (ICIP), pp 3464–3468. https://doi.org/10.1109/ICIP.2016.7533003
18. Wojke N, Bewley A, Paulus D (2017) Simple online and realtime tracking with a deep association metric. In: 2017 IEEE international conference on image processing (ICIP), pp 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962
19. Mekkayil L, Ramasangu H (2018) Object tracking with correlation filters using selective single background patch. arXiv:1805.03453
20. Bolme DS, Beveridge JR, Draper BA, Lui YM (2010) Visual object tracking using adaptive correlation filters. In: 2010 IEEE computer society conference on computer vision and pattern recognition, pp 2544–2550. IEEE
21. Vergés-Llahí J, Ar J, Sanfeliu A (2001) Object tracking system using colour histograms
22. Zivkovic Z, Krose B (2004) An em-like algorithm for color-histogram-based object tracking. In: Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition, 2004. CVPR 2004., vol 1. https://doi.org/10.1109/CVPR.2004.1315113
23. Li X, Yin H, Zhou K, Zhou X (2020) Semi-supervised clustering with deep metric learning and graph embedding. World Wide Web 23:781–798
24. Deng B, Jia S, Shi D (2019) Deep metric learning-based feature embedding for hyperspectral image classification. IEEE Trans Geosci Remote Sensing 58(2):1422–1435
25. Cao R, Zhang Q, Zhu J, Li Q, Li Q, Liu B, Qiu G (2020) Enhancing remote sensing image retrieval using a triplet deep metric learning network. Int J Remote Sens 41(2):740–751
26. Nvidia (2023) TensorRT. https://developer.nvidia.com/tensorrt. Accessed: 18-April-2023
27. Abadi M, Agarwal A, Barham P, et al (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. https://www.tensorflow.org/
28. Ultralytics (2023) YOLOv8. https://github.com/ultralytics/ultralytics. Accessed: 18-April-2023
29. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520
30. Hermans A, Beyer L, Leibe B (2017) In defense of the triplet loss for person re-identification. arXiv:1703.07737

31. Wang L, Shi J, Song G, Shen I-f (2007) Object detection combining recognition and segmentation. In: Yagi Y, Kang SB, Kweon IS, Zha H (eds) Computer vision – ACCV 2007, pp 189–199. Springer, Berlin, Heidelberg
32. Li W, Zhao R, Xiao T, Wang X (2014) Deepreid: Deep filter pairing neural network for person re-identification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 152–159
33. Herzog F, Ji X, Teepe T, Hörmann S, Gilg J, Rigoll G (2021) Lightweight multi-branch network for person re-identification. In: 2021 IEEE international conference on image processing (ICIP), pp 1129–1133. IEEE
34. Li D, Chen S, Zhong Y, Liang F, Ma L (2022) Dip: Learning discriminative implicit parts for person re-identification. arXiv:2212.13906
35. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929
36. Zhang S, Yin Z, Wu X, Wang K, Zhou Q, Kang B (2021) Fpb: feature pyramid branch for person re-identification. arXiv:2108.01901