HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

Person Monitoring on Edge Devices using Computer Vision

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

Major: Information Technology

Supervisor: Dr. Dang Tuan Linh _____

Signature

Department: Computer Engineering

School: School of Information and Communications Technology

HANOI, 08/2023

ACKNOWLEDGMENT

Firstly, I would like to express my gratitude to Dr. Dang Tuan Linh for all of his assistance, not just in helping me finish this thesis but also during my four years of university study. I have learned so much from him and those values are extremely valuable to me, which have greatly assisted me in both work and study. Additionally, I also want to sincerely thank Mr. Hung and Ms. Quynh, without whom it's likely I would not have been able to successfully complete this thesis. I am also grateful to other friends, seniors, and contributors, whether in small or large ways, tangible or intangible, for their contributions to this completed thesis today. Finally, I would like to thank my family for always supporting me unconditionally.

ABSTRACT

In today's fast-paced world, effective management and analysis are crucial for maintaining security and enhancing business productivity, particularly as the number of enterprises and the size of enterprises rise. According to a recent report from the General Statistics Office of Vietnam, between 2016 and 2019, there was an average annual increase of 9.8% in the number of enterprises, which is higher than the average annual growth rate of 8.1% observed between 2011 and 2015. Moreover, the behavior of employees can be difficult to manage, and employers can increase productivity if they have valuable information from human behavior. Therefore, it is necessary to implement modern technology, specifically AI, into monitoring systems in order to reduce costs and increase productivity.

However, the majority of current AI implementations rely on centralized servers, making scaling difficult. This thesis proposes a novel AI module that can be installed on edge devices, as a means of overcoming this obstacle. Human detection, human tracking, and human feature extraction are the three primary components of the proposed module. All of these components are directly executed on edge devices. This AI module can be utilized to monitor individuals and collect data that can be used to enhance the productivity of businesses.

I aim to achieve efficient human management and analysis by implementing AI models on edge devices that are readily scalable. The algorithms utilized in this thesis have been successfully implemented on Jetson Nano devices with low computational capability while maintaining above 10 FPS for less than seven people in a single frame. A prototype of this module has been put into practical use and examined in room 405, B1 building at Hanoi University of Science and Technology. The proposed module has the potential to revolutionize office human resource management and analysis, thereby enhancing office security and productivity while reducing costs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
AI	Artificial Intelligence
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
EER	Equal Error Rate
FPS	Frames Per Second
GPU	Graphics Processing Unit
ID	Identification
IoT	Internet of Things
IoU	Intersection over Union
mAP	mean Average Precision
NMS	Non-maximum Suppression
Re-ID	Re-identification
ROI	Region Of Interest
USB	Universal Serial Bus

CHAPTER 1. INTRODUCTION

In the first chapter, I will discuss the motivation leading to the birth of this thesis and the state of current solutions. Based on these, I will introduce my proposed solution for human monitoring on edge devices based on computer vision.

1.1 Motivation

Nowadays, managing human resources is not an easy task for businesses. Every year, there is not only an increase in the number of enterprises but also an increase in the number of large-size enterprises. Between 2016 and 2019, there was an average annual increase of 9.8% in the number of enterprises, which is higher than the average annual growth rate of 8.1% observed between 2011 and 2015 [1]. With the increasing size of businesses, it is impractical for employers to manage people by traditional methods because they are time-consuming and inefficient. These problems have forced businesses to find a new way to manage human resources. Today, with the fast-paced developments of technology, especially AI, more and more businesses digitize their processes and replace inefficient, human-operated tasks with machines and algorithms. There is a high demand [2] for a system that can have the timekeeping ability, maintain security, and improve productivity. If there is a system that can solve human monitoring problems and provide useful information to boost the productivity of businesses, it can be widely used in many more areas and places like supermarkets, shopping malls, or galleries.

Modern technology provides employers with plenty of more efficient ways to manage human resources [3]. They can be identifications using magnetic cards or fingerprints each time an employee enters or exits the office [4]. Another more advanced option is to use facial recognition algorithms based on deep learning to authenticate employees each time they enter or leave the office. However, no matter which method is used, they have their limitations. Most of the widely used management systems in Vietnam such as Base HRM [5] or AMIS HRM [6], only focus on timekeeping and security problems, they don't provide any statistics of people inside the building which bring a lot of value to the company. Moreover, deep learning-based methods like automatic facial authentication with great using frequency require a strong and expensive centralized server, and they are hard to scale up the system. Most of the edge devices in these systems only send images to the server and the deep learning models will be executed in the server. This will put a lot of computing pressure on the server when the number of edge devices increases.

On the other hand, with the rapid development of computing infrastructures, there is an increase in the utilization of edge devices across various industries because of the advantages they offered. They are portable, easy to scale up, and cheaper than normal computers [7]. Moreover, nowadays advanced lightweight algorithms enable costly computing methods like deep learning models to be implemented on resource-constrained devices. These lightweight models have high accuracy while having a very light architecture [8], [9]. Taking advantage of the availability of surveillance cameras in offices and supermarkets, these edge devices can be quickly and economically integrated to create comprehensive monitoring systems.

There is a significant demand for a human monitoring system that can automatically monitor individuals and provide valuable statistical data on everyone within a building. Additionally, it is important that the computational workload of this system be distributed among edge devices to prevent overloading the system. Therefore, this study researches and develops an automatic human monitoring module based on computer vision that can be implemented on resource-constrained devices. These edge devices then can be used in a management system.

1.2 Objectives and scope of Thesis

Section 1.1 shows various ways of managing human resources that are no longer appropriate for current situations or have their own weaknesses. The first objective of this thesis is to build a stand-alone remote human monitoring module, implemented on edge devices, Jetson Nano. The module consists of three main components: human detection, human feature extraction, and human tracking. The proposed module can detect humans, track them, and give each individual identity features based on their outlook. The output of this module can later be integrated into a bigger management system to do timekeeping, maintain security, and increase business productivity by analyzing data provided by multiple edge devices.

Moreover, to lower the workload of the server, the second objective is to implement deep learning models directly on edge devices. This module takes the input source from a connected USB camera. The models are tailored or optimized to be suitable for deployment on Jetson Nano which has limited computing resources. To achieve faster inference speed, this thesis also presents a customized feature extraction model specifically for resource-constrained devices.

The scope of this thesis is limited to a working prototype implemented on Jetson Nano of a human monitoring module constructed from lightweight models and algorithms. Due to cost constraints, there are only a total of three Jetson Nano devices, one is placed at the entrance, and the remaining are placed inside two

different rooms. The operational area of the system is limited to the observed areas of implemented cameras. Only people who have been registered in the system are allowed to enter the room. Jetson Nano uses attached cameras to monitor rooms. AI models are executed directly on Jetson Nano to reduce the load on the server. Collected information is sent from multiple Jetson Nano devices and processed by the server. The results from the server can be observed in an application developed by other members of my research group.

1.3 Tentative solution

Based on the presented objectives in Section 1.2, the work composes of two main parts.

In the first part, based on researched technologies, the pipeline of the module will be proposed. The module will consist of three main components: human detection, human feature extraction, and human tracking which will be implemented in Jetson Nano. The human detection model will be carefully selected among state-of-the-art models based on the criteria of accuracy and lightness so that it can be run on Jetson Nano. In addition, a tracking algorithm will also be implemented on Jetson Nano to utilize the output of the feature extraction model and track humans from the input source.

The second part includes building a tailored, lightweight architecture of the feature extraction model specifically for limited computing resources devices and benchmarking the proposed model with current state-of-the-art models on mAP and Rank-1 metric over a well-known public dataset CUHK03-labeled. Moreover, all the models will be further optimized to be faster on Jetson Nano. A prototype will be deployed in practice for testing purposes.

1.4 Contributions

This thesis contains two main contributions as follows:

1. A full pipeline of the module for human monitoring tasks on edge devices is proposed. The module contains three main components: a human detection model, a feature extraction model, and a tracking algorithm.
2. This thesis also proposed a tailored lightweight feature extraction model to make it practical to run on resource-constrained devices.

1.5 Organization of Thesis

The remaining of this thesis will be organized as follow:

Chapter 2 will discuss some related works and foundation theories of existing detection, feature extraction, and tracking algorithms. The proposed module will

be built based on these investigations.

Chapter 3 will introduce the proposed module on edge devices. This module contains three main components: human detection model, human feature extraction model, and human tracking algorithm. The flow of this module and details about each component will be presented. Moreover, the hardware implementations and utilization of frameworks are also provided in this chapter.

Chapter 4 will present evaluations of deep learning models and some results of the deployed system. The settings and datasets used for evaluations are also provided. Based on obtained experimental results, discussions will be made.

Chapter 5, the final chapter, will provide a summary of the achieved results. Additionally, suggestions for future improvements will also be included.

CHAPTER 2. LITERATURE REVIEW

Chapter 1 has discussed the problems which lead to the motivations of this thesis. It has also presented the objectives, the tentative solutions, and the contributions of this thesis. There are two main parts in this chapter. They are presentations about (i) related works in Section 2.1, and (ii) the foundation theory in Section 2.2. Specifically, in Section 2.2, models, and algorithms contributing to the making of the lightweight module on edge devices will be introduced in detail. For the purpose and scope of the thesis, four main components that make up the proposed human monitoring module on edge devices will be introduced including: (i) object detection with YOLOv5 in Section 2.2.1, (ii) object tracking algorithms including SORT and DeepSORT in Section 2.2.2, (iii) deep metric learning with particular emphasis on the hard triplet loss function in Section 2.2.3, and (iv) MobileNetV2 architecture in Section 2.2.4.

2.1 Related works

2.1.1 Person re-identification

One of the main objectives of the proposed module is re-identifying persons across multiple cameras. Person re-identification can be considered a person retrieval problem. A person Re-ID system is used to recognize individuals who move out of a camera's view and then reappear in another camera's view or current camera view at another timestamp. For this reason, it can be used to track specific entities across multiple cameras. Facial features are often used to identify a person. However, when the image quality at the end camera is poor, identification based on faces becomes almost impractical. To overcome this problem, many researchers have used outlook characteristics to re-identify a person. Despite its widespread use in security, traffic, and commercial settings, person re-ID systems face many challenges such as variations in lighting, viewing points [10], complex backgrounds, and outfits worn by individuals [11]. In early research, hand-crafted features are constructed from clothes or body shapes [12], [13]. One of their greatest limitations is that they don't generalize well in a new environment as they are designed based on a specific set of assumptions. However, with the advancement of technology, deep learning has been widely studied and applied to extract features automatically. It shows inspiring results on different benchmarks and may be more robust in practice.

Person Re-ID is a crucial component of a human monitoring system, it must be used to re-identify the person who moves across cameras. Nevertheless, deploying the Re-ID system based on deep learning with many cameras can be complex,

costly, and problematic in scaling.

2.1.2 Edge computing in AI

AI is becoming more and more important and appearing more often in our human daily lives. In recent years, the development of IoT [14]–[16] has promoted the development of AI under a mass amount of provided data from end devices. Because of these huge amounts of data generated from IoT, it is a big challenge for traditional centralized cloud servers to process all of this data. Moreover, AI applications also require a vast amount of computing which will put a burden on centralized servers. At the same time, the rapid advancement of computing infrastructures has made edge computing prevalent in many fields [17]. Furthermore, the breakthrough of lightweight deep learning models developed in recent years [18] has enabled edge devices to have the capability to conduct computing for AI tasks at the end level which can alleviate the working burden of a centralized server [9]. In addition, they are also portable, easy to scale up, and cheaper than normal computers.

For these reasons, building a system capable of conducting deep learning-based person Re-ID at the edge level becomes extremely important and practical nowadays. A human monitoring system with multiple cameras can use edge computing to alleviate the computing stress on the server.

2.2 Foundation theory

2.2.1 Object detection

The first phase of my proposed module is detecting humans in the input frame. Object detection is an important task in computer vision that identifies specific objects within an image. This method is extensively utilized in numerous areas, including medicine, robotics, and surveillance. One of the famous traditional object detection algorithms is the use of the histogram of oriented gradients with support vector machine [19]. With the advancement of AI technology, object detection models based on deep learning have outperformed traditional ones. A model named Regions with CNN Features (R-CNN) [20] has substantially improved the performance of object detection tasks. This algorithm has used convolutional neural networks (CNN) [21] in combination with a region proposition algorithm which helps it to accurately identify objects in images and classify them. There are many later improvements based on R-CNN such as Fast R-CNN [22] and Faster R-CNN [23]. Despite showing inspiring accuracy, these two-stage architectures have slow inference speeds and can not be implemented to solve tasks that required low latency. In contrast, You Only Look Once (YOLO) [24], a real-time single-stage model, has

shown a great performance in both accuracy and latency. YOLO will divide the image into a grid of cells and predict the detection score, class probability, and bounding box coordinates for each cell. YOLOv5 [25], an improved version of the original YOLO, has achieved better accuracy and had a very short inference time which makes it become an ideal selection for real-time applications.

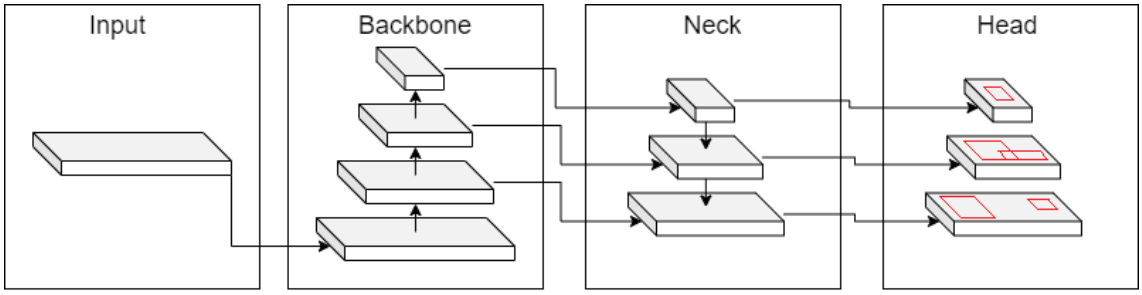

Figure 2.1: The overview architecture of YOLOv5 [25].

YOLOv5 [25] is a model in the YOLO family. It is a single-stage object detection model that has three main parts: backbone, neck, and head as illustrated in Figure 2.1. Especially, YOLOv5n, the nano version of YOLOv5 is very lightweight, it has 1.9 million parameters which makes it a suitable model for edge computing. Among all versions in the family of YOLO, YOLOv5n is the most lightweight and is suitable for deployment on edge devices.

The backbone is a pre-trained CNN model which is used to extract features from input images. The feature extraction process will gradually decrease the spatial resolution of the tensor while increasing its number of channels. Obtaining the output from different layers gives us features from varied spatial resolutions. Moreover, the model neck will combine features from different scales from the backbone to build feature pyramids. This will give the model the ability to be robust with objects of different sizes and scales in images. Finally, from the output of the neck, the model head will take responsibility for predicting classes of objects, corresponding confidence scores, and bounding boxes.

Because of its lightweight and accurate characteristics, YOLOv5n is utilized in my module which is deployed on resource-constrained devices to detect humans from input frames.

2.2.2 Object tracking

After a person is detected in the frame, that individual needed to be tracked over time in that camera. Object detection only helps us to detect desired objects in each distinct frame. To match detected bounding boxes from consecutive frames to only one entity, an object tracking algorithm is needed.

Object tracking is the process of following an object's movements and preserving its identity, even if its appearance or motion changes. Traditional methods for object tracking, such as Multiple Hypothesis Tracking (MHT) [26], create multiple tracks for each object and use a data association algorithm to select the most probable track at each time step. However, recent advances in deep learning and object detection models with neural networks have led to the rapid development of modern tracking algorithms. These algorithms are the cooperation between object detection models and motion models. Motion models use predicted bounding boxes of detection models at an interval of time step as input to estimate the motions of objects and construct entities. For instance, SORT (Simple Online and Realtime Tracking) [27] is a lightweight algorithm that uses the Kalman filter to predict the state of each object in each frame and the Hungarian algorithm to associate the predicted states with the detected objects in the current frame to decide which detected objects belong to which tracked objects. Even though, SORT only considers the geometric characteristics of objects. With further improvement on weaknesses of SORT, DeepSORT [28] incorporates a Siamese network that has been pre-trained to distinguish between individuals and add visual information to the estimation process, leading to inspiring improvements in case of occlusion where SORT fails to function. Other appearance-based methods, such as correlation filters [29], [30] and color histogram-based trackers [31], [32], have also been extensively employed for object tracking.

a, SORT

SORT is a detection-based tracking algorithm. SORT comprises four main components: detection, propagating states of tracked objects into future frames, associating existing objects with detected bounding boxes at current frames, and managing the life span of objects.

SORT takes the output results of the detection model as its input. The predicting quality of the detection model affects greatly the accuracy of SORT. Therefore, the advancement of deep learning with CNN has directly improved object detection models and indirectly boosted the performance of this tracking algorithm.

Moreover, SORT uses an estimation model to propagate states of tracked objects into future frames by using a linear constant velocity model. The model will estimate the position and velocity of each entity. When a detected bounding box is associated with a tracked object, the state of this object will be updated by this bounding box through the Kalman filter [33]. If no detection is associated with this tracked object, its state will be directly estimated by the linear constant velocity model without any

rectifications.

To assign detections to previously identified targets, SORT will first estimate the state of each target in the current frame which is its bounding box. After that, SORT computes the distance metric between estimated bounding boxes and detections from the object detection model. This metric is IoU and the assignment will be solved by the Hungarian algorithm. This process helps the algorithm determine which detections should be assigned to which targets based on their respective bounding box geometries. Moreover, a minimum threshold is set for the assignment in order to not match tracked objects with detections that are too different from estimated states. This can prevent fault caused by short-term occlusion.

When objects appear or disappear from the image, the algorithm creates or removes identities as necessary. A new tracker is created when there is a detection that can not be matched with any existing objects. This tracker is then initialized using the detected bounding box and the velocity is set to zero. Furthermore, the newly tracked object is supervised for a few next consecutive frames where it needs to be matched with detections from the object detection model to accumulate confidence and become a new identity officially.

b, DeepSORT

While SORT achieves an overall good performance, it still faces many problems especially a high number of identity switches in scenarios containing occlusions, crowded scenes, and fast-moving objects. For that reason, DeepSORT has come up with the idea to combine visual descriptors beside the velocity and motion of objects to improve data association.

Visual descriptors in DeepSORT are feature vectors that are extracted from patches in the image. These patches are cut from the original image based on the information of predicted bounding boxes. Each of these regions will be fed through a deep neural network to extract the visual feature vectors. After that, these feature vectors are normalized to obtain final visual representations of detections. Distances between tracked objects and new detections are calculated by the cosine similarity.

DeepSORT has better accuracy compared to SORT but it has a tradeoff with inference time. Despite being not as fast as SORT, DeepSORT is still a very fast algorithm which makes it practical to be implemented on edge devices. The breakthrough concept of enhancing tracking precision by integrating both motion and visual information is incorporated into my tracking algorithm in the monitoring module to track people moving in the sight of a single camera.

2.2.3 Deep metric learning

Metric learning is a type of machine learning that involves learning to present data points to a metric space. In this space, the distance between similar data points is expected to be small, and dissimilar data points are expected to be far apart. Metric learning can be applied to my proposed module by projecting ROI from multiple cameras or the same camera but at different timestamps of the same person to a cluster of representations close together.

Deep metric learning is a special type of metric learning which utilizes deep neural networks to automatically learn the mapping from the input space to the metric space from the data. This mapping is achieved by using a distance-based loss such as contrastive loss [34] or triplet loss [35]. After learning a representation, it can be utilized for other tasks such as classification [36] or clustering [37]. Many practical applications, such as face recognition and image retrieval [38], have benefited from the effectiveness of deep metric learning.

Triplet loss [35] is one of the most popular loss functions in deep metric learning. It has the same objective as other types of distance-based loss functions which is shortening distances between similar data points and enlarging those of dissimilar data points. While contrastive loss [34] directly optimizes the above objective, triplet loss achieves that goal in an indirect way. It motivates dissimilar pairs to have distances larger than any similar pairs by a chosen margin which is illustrated in Figure 2.2. Triplet Loss tries to minimize the distance between an anchor and a positive data point which have the same label to be smaller than the distance between the anchor and a negative data point which have a different label plus the margin constant.

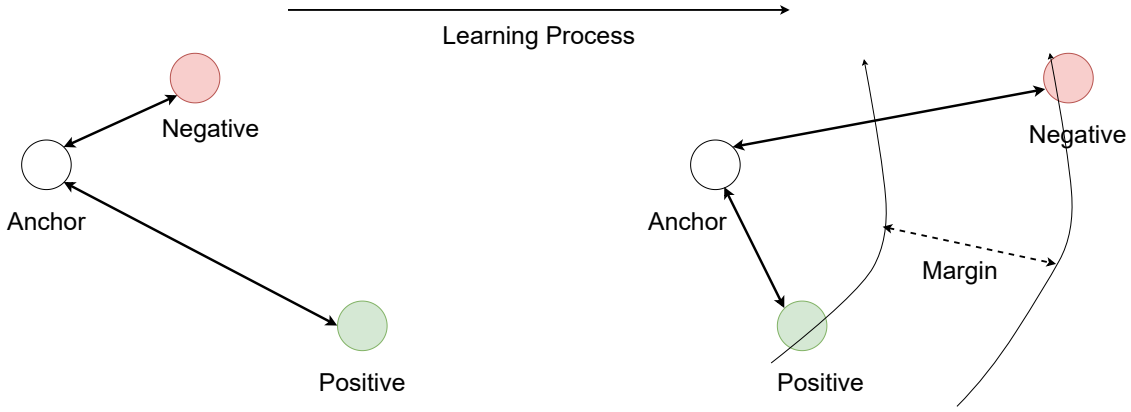

Figure 2.2: The learning process of triplet loss.

The triplet loss function can be defined as below:

$$Loss = max(d(a, p) - d(a, n) + m, 0) \tag{2.1}$$

In which:

- d is a distance metric between two representations such as Euclidean distance or cosine similarity.
- a is the anchor sample.
- p is the positive sample that has the same label as the anchor.
- n is the negative sample that has a different label as the anchor.
- m is the margin constant.

The triplet loss method doesn't forcibly push the anchor and positive samples to be the same data point in the target space as the contrastive loss. It enables triplet loss to tolerate some intra-class variance which reflects the fact that the samples in the same class are not actually identical and that there are some outliers in datasets. Moreover, the triplet loss also doesn't optimize in an exaggerated way as contrastive loss which shortens the distance between similar pairs as much as possible. On the contrary, triplet loss only requires the distance between positive pairs to be lower than negative pairs by some margin value.

In addition, hard triplet loss is the triplet loss with the hard mining strategy [39]. Mining strategy is the way triplets are constructed and it helps the optimization process to converge faster and achieve better performance. In a mini-batch, this mining strategy will construct a triplet for each sample in the mini-batch by creating the corresponding hardest positive pair and hardest negative pair. In other words, it will choose the positive sample which has the maximum distance from the anchor, and the negative sample which has the minimum distance to form the triplet. All data points that were assessed are contained in a single mini-batch.

With mentioned capabilities, hard triplet loss plays a vital role in building my proposed feature extraction model. It gives the model the ability to re-identify individuals across cameras.

2.2.4 MobileNetV2

MobileNetV2 [9] is a lightweight neural network architecture designed for resource-constrained devices. MobileNetV2 significantly reduces the number of parameters and inference time while still achieving good accuracy. The author has introduced new bottleneck modules with inverted residual architecture.

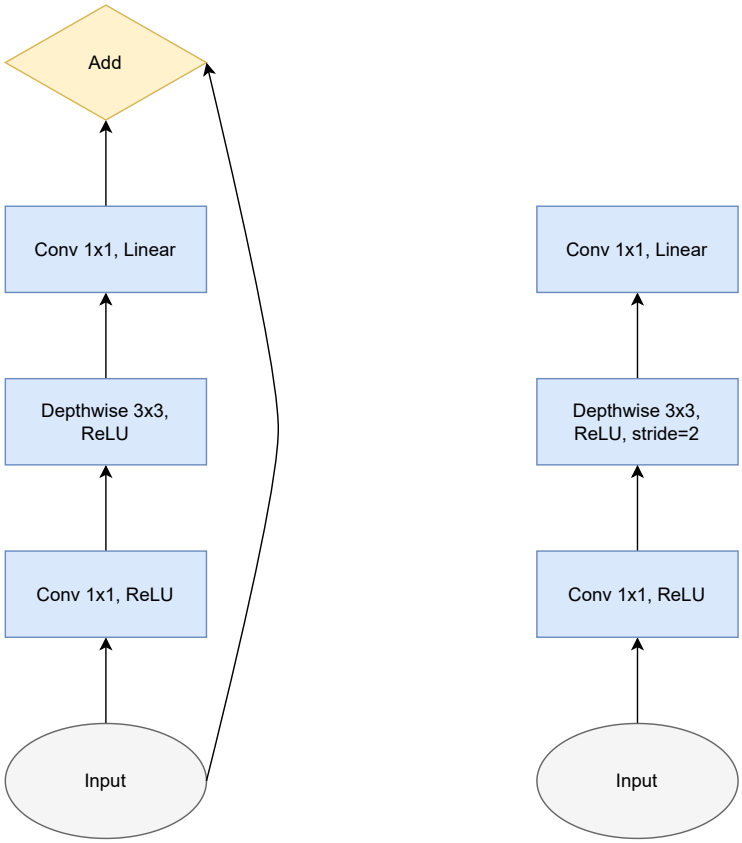

Figure 2.3: Bottleneck modules of MobileNetV2.

Figure 2.3 shows bottleneck modules of MobileNetV2 with residual connection on the left one and depth-wise layer with stride equal to two on the right one to shrink the representations. These modules take a low-dimensional tensor as input. After that, this tensor is enlarged to a high-dimensional space through a 1×1 point-wise convolution. This point-wise convolutional layer computes linear combinations of input channels and produces an output tensor with a preferred number of channels. Next, a 3×3 depth-wise convolution is applied followed by a rectified linear unit (ReLU) activation function. This convolutional layer operates by applying a distinct filter to each input channel, after which all resulting output tensors are combined into a single tensor. Finally, another 1×1 point-wise convolutional layer is applied to project the tensor back to a low-dimensional space. In addition, in the left module, a residual connection is used to connect two low-dimensional representations. This design improves backpropagation and costs less memory than usual residuals. On the other hand, the second module with a stride equal to two in the depth-wise layer helps to shrink the height and width of the tensor gradually. These architectures help the model to reduce a lot of parameters while maintaining accuracy. Therefore, models which are customized from MobileNetV2 have the advantages to be deployed on edge devices with limited computing capability.

This chapter has shown a detailed discussion of foundation theories. YOLOv5 is a lightweight object detection model which is appropriate for applications implemented on edge devices while retaining decent accuracy. Hard triplet loss has shown its essentiality in the problems of person Re-ID. SORT is a simple but fast tracking algorithm. On the other hand, DeepSORT proves its strength in problems that need high accuracy. In the upcoming Chapter 3, I will discuss my development of an automatic person monitoring AI module that can be installed on edge devices. This development was informed by research results and an analysis of relevant theories and algorithms.

CHAPTER 3. METHODOLOGY

In this chapter, using researched foundation theories in Chapter 2, I will present (i) the overview of the AI module as well as an outline of how it is integrated into a larger management system, (ii) the proposed AI module in detail, (iii) hardware implementation, and (iv) the utilization of frameworks.

3.1 Overview

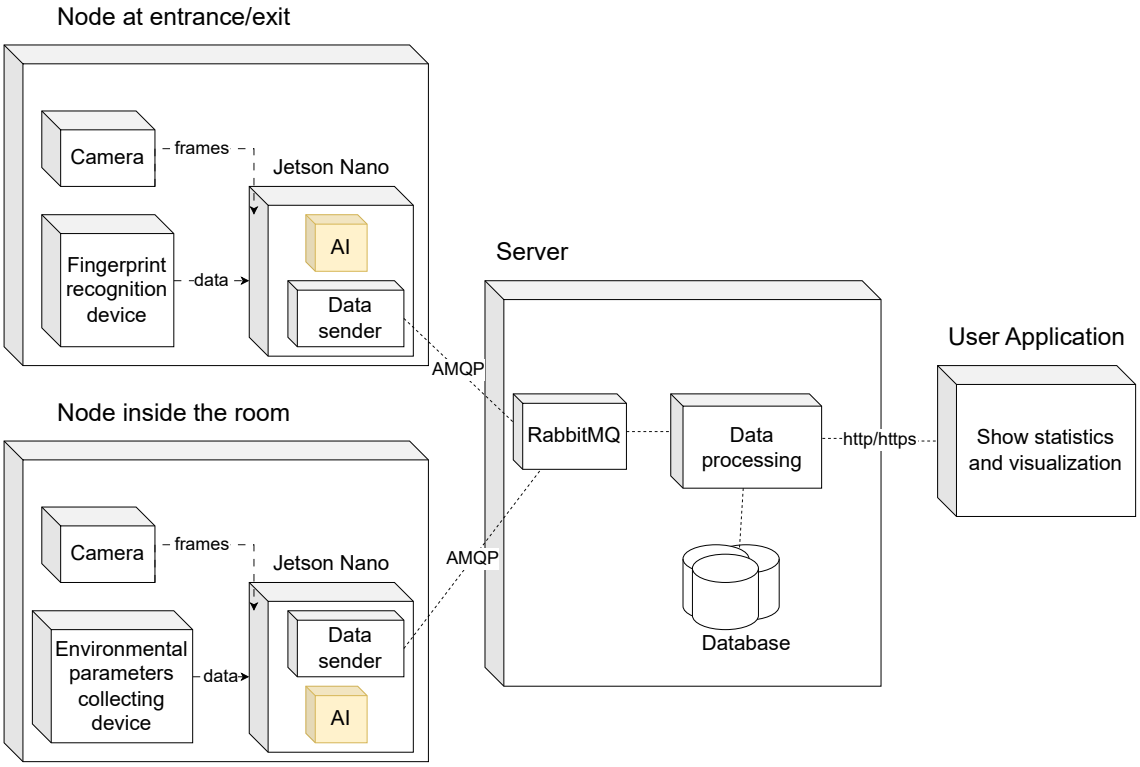

Figure 3.1: The overview of a full end-to-end system.

Figure 3.1 shows the overview of a full end-to-end human monitoring system. This system has peripheral devices like cameras to capture frames or sensors to collect environmental parameters. A small, single-board, low-power computer, Jetson Nano is in charge of edge computing, it receives data from sensors and sends them to the server. Furthermore, it takes the stream from the attached camera and conducts all the AI computing tasks, and then sends the processed information to the server to store. Jetson Nano devices are placed at the entrance and in multiple positions inside different rooms to monitor. This system also has a server to store data sent from Jetson Nano and handle requests from user applications. Finally, the end-user application can show statistics and do visualization based on information received from the server.

The scope of this thesis only focuses on the development of the AI module on

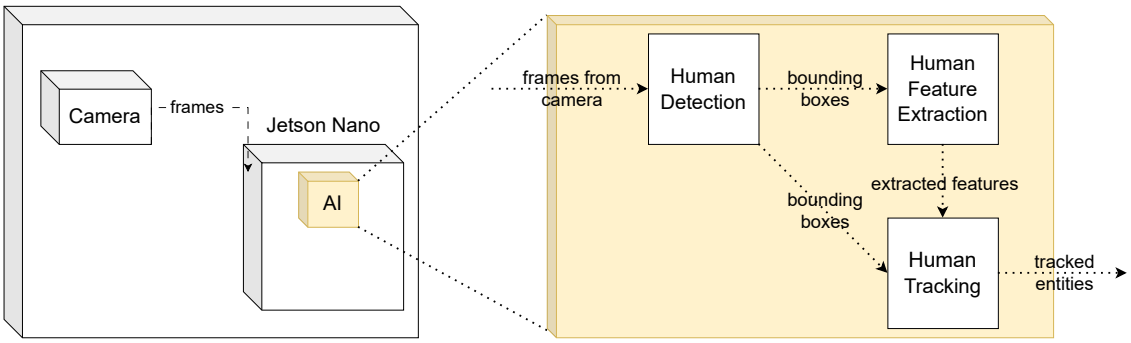

Figure 3.2: The overview of the AI module on edge devices.

edge devices as shown in Figure 3.2 and other parts of this system are developed by other members of my research team. Jetson Nano receives input frames captured by the attached camera. Then, the object detection model detects humans on these frames and outputs the corresponding bounding box for each person. Next, the ROI of each person on the image is cut based on the bounding box and fed to the feature extraction model. This model takes images as input and extracts the feature of each image as a vector with 224 dimensions. These vectors are visual descriptors of cut ROIs. Finally, the geometric information provided by bounding boxes and visual characteristics provided by extracted features is combined and fed to the tracking algorithm. Then, this tracking algorithm tracks entities throughout consecutive frames.

The input of the AI module includes:

- Stream of raw frames from the camera.

The output of the AI module includes:

- Tracked entities in the input stream include:
  - Local identification number represents a unique person.
  - Bounding box location of that person in the current frame.
  - Visual feature for the bounding box. Although these visual features are sent from independent devices, they can later be used by the server to map different entities from different devices. Therefore, people who move from one camera to another camera, from one room to another room, can be tracked.

Moreover, Jetson Nano also conducts other works such as sending raw frames from the camera and parameters of environmental conditions to the server.

3.2 The proposed AI module

Section 3.1 has introduced the overview of the pipeline of the AI module implemented on edge devices. Section 3.2 will provide a detailed explanation of each component in the AI module.

3.2.1 Human detection

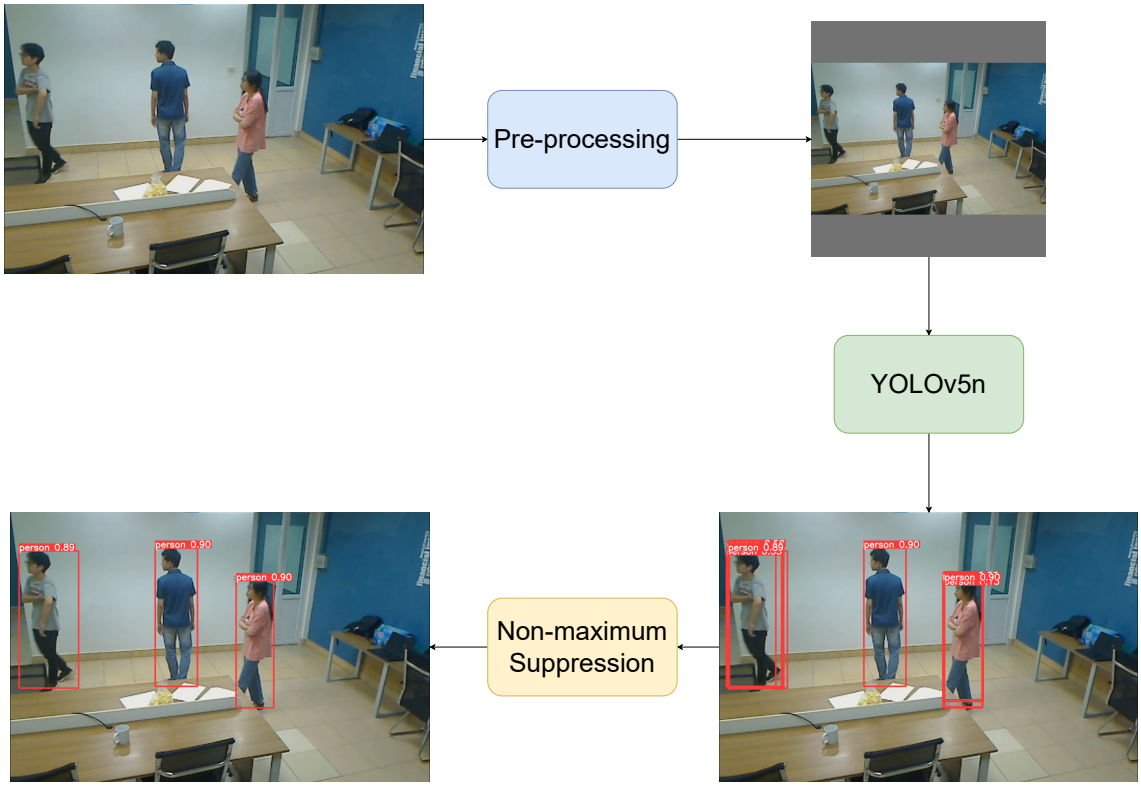

Figure 3.3: The process of human detection.

The human detection model takes input frames and detects humans on them. Figure 3.3 illustrates three parts of the human detection process. Before feeding inputs into the human detection model, they are first pre-processed. After obtaining bounding boxes from the model, they will be processed by the non-maximum suppression algorithm to get the final outputs.

Each output bounding box will have the following components:

- Coordinates of the box in the current frame. The coordinates contain four elements: $xmin$, $ymin$, $xmax$, and $ymax$.
  - $xmin$ is the left bound of the box.
  - $ymin$ is the upper bound of the box.
  - $xmax$ is the right bound of the box.
  - $ymax$ is the lower bound of the box.

- The class of the box. In this case, this element is trivial because there is only one class in the detection model.
- The confidence score. This indicator shows us how confident the model is in the prediction. The confidence score ranges from 0 to 1. The higher the confidence score, the more confident the model is in the prediction.

Outputs from the detection model are utilized by the feature extraction model and the tracking algorithm. YOLOv5n, the nano version of YOLOv5, is used as the object detection model because of its lightweight and low latency characteristics as mentioned in Section 2.2.1. I use the pre-trained weights of YOLOv5n, which has already been trained with the COCO dataset [40]. I choose to use this pre-trained instead of training from scratch because this huge dataset has already contained the $person$ class which is the objective class of the desired model.

a, Pre-processing

Before feeding images into the detection model, they need to be pre-processed to transform all images of different resolutions into the same format. The pre-processing method must be exactly the same as during model training. The desired size is 384×384 and raw images are resized to this resolution. I choose this resolution instead of the original resolution 640×640 of YOLOv5n to reduce the computing cost and reduce latency. Computing on the original resolution is a burden for the edge device.

First of all, the larger dimension of the raw image is resized to 384 and the other dimension is resized with the same ratio. This makes the length of the second dimension to be lower or equal to 384. Therefore, to achieve the desired length of 384, both sides of the second dimension are padded equally with gray color of value $(114, 114, 114)$. Moreover, all the pixel values are divided by 255 to normalize the image to the range $[0, 1]$.

b, Detecting with YOLOv5n

As mentioned before, the pre-trained weights of YOLOv5n were used. These weights have been pre-trained with the COCO dataset [40]. I only use the $person$ class and ignore all other classes. This version of pre-trained weights was trained with the input size of 640×640. Although this input size is not my desired size which is 384×384, this pre-trained model can still be utilized well because of the pyramid architecture of YOLOv5n which enable it to be robust with objects of different size. Moreover, input images of arbitrary resolutions can also be computed because of the CNN architecture. Due to limited computing resources on edge devices, I choose to infer the model on a smaller resolution to lower the computation

cost. Moreover, this model is further converted into a TensorRT engine for high-performance deep learning inference on Jetson Nano. Its weights are quantized from 32-bit floating points to 16-bit floating points for a faster computation speed.

After feeding the processed input into the model, many detected bounding boxes overlap with each other. Therefore, a post-processing algorithm is needed to remove redundant bounding boxes and it is called Non-maximum suppression which will be presented in the next part.

c, Non-maximum suppression

The detection model produces a lot of overlapping detections. A single individual in the image can have more than one bounding box. One object with multiple corresponding bounding boxes can cause trouble at a later stage. All unnecessary bounding boxes need to be filtered out and only one bounding box is remained. This remaining bounding box is chosen by the non-maximum suppression algorithm. NMS helps to take one best bounding box out of many overlapping bounding boxes. It chooses a pair of overlapping bounding boxes by IoU threshold which is described in Section 4.3.1 and removes the one with a lower confidence score as shown in Figure 3.4. The process is repeated until there is no pair of overlapping detections left. Through experiments, the IoU score was chosen to be 0.45 because this threshold showed the best result. An excessively high threshold may cause unnecessary bounding boxes to persist, whereas an excessively low threshold may lead to the merging of bounding boxes belonging to two different individuals who are standing next to each other. Besides, before running NMS, all bounding boxes that have a confidence score lower than 0.45 are first filtered out to reduce the workload for NMS.

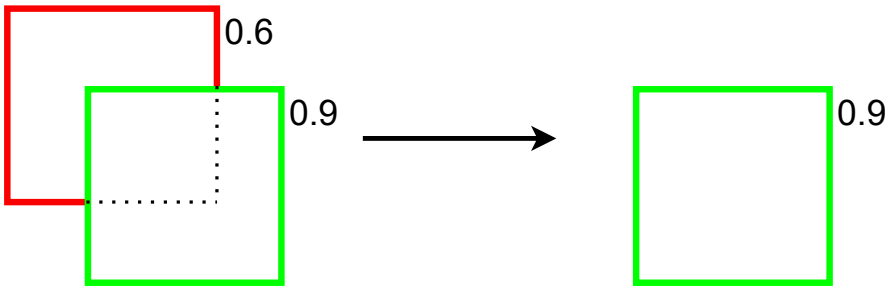

Figure 3.4: NMS filters out the bounding box with the lower confidence score in a candidate pair of overlapping bounding boxes.

After running NMS, we obtain bounding boxes with no high overlapping. These final bounding boxes are utilized in later parts: Human feature extraction in Section 3.2.2 and Human tracking in Section 3.2.3.

3.2.2 Human feature extraction

Feature extraction model plays a crucial role in the human monitoring system. It provides the system with the ability to Re-ID across different cameras. Moreover, it also improves the quality of the tracking algorithm within independent cameras. The human feature extraction model takes the pre-processed ROI cut from the raw image based on the detected human bounding box as input and it outputs a 224-dimensional feature vector as shown in Figure 3.5. These feature vectors are sent to the server to store and conduct cross-camera re-identification. The server assesses whether the vectors received from various edge devices belong to the same individual or not. In addition, these features are also used by the human tracking algorithm in Section 3.2.3.

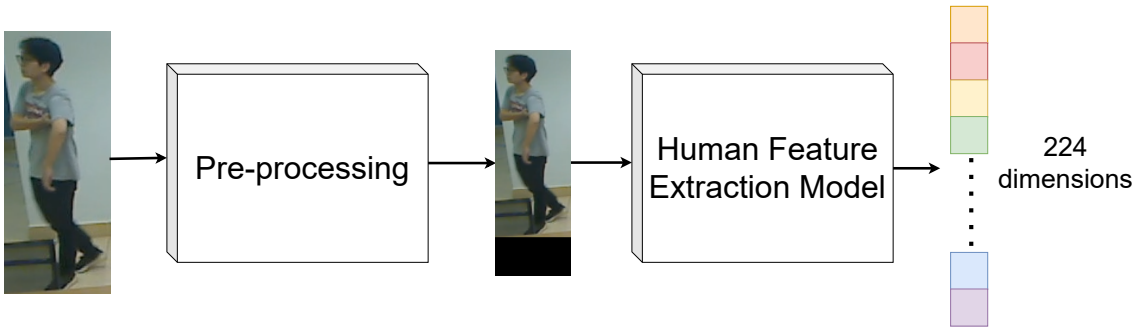

Figure 3.5: The process of human feature extraction.

This thesis proposes a customized version of MobileNetV2 as the backbone of the feature extraction model to create a lightweight architecture suitable for edge devices with limited resources. Since the computation time of the human feature extraction model increases with the number of people in the frame, this model needs to be very lightweight. Therefore, I use a customized version of the original MobileNetV2 architecture to further reduce latency.

The proposed model is trained on public datasets and subsequently deployed on edge devices. Furthermore, this model is also converted to TensorRT format as the object detection model for high-performance inference on Jetson Nano. Despite this, the trained weights still use 32-bit floating points to maintain the accuracy of the model. This is necessary because the output features vector needs to be accurate in order to accurately match individuals across multiple cameras.

a, Pre-processing

Before feeding the images into the model, they need to be pre-processed to bring images of different resolutions to the desired size 192×64. First of all, the width of the input image is resized to 64, and the height of the image is resized correspondingly with the same ratio. Next, if the height of the resized image is

larger or equal to 192, this dimension will be resized to shrink it to 192. Conversely, if the height of the resized image is less than 192, black pixels with a value of 0 will be added to the bottom of the image until the height is 192. Moreover, the pixel values of the pre-processed image are then divided by 255 to map the value range to $[0, 1]$.

b, Tailored MobileNetV2 as backbone

Original MobileNetV2 is tailored to further reduce the number of parameters. Instead of using the original MobileNetV2, a tailored model which only contains a portion of MobileNetV2 as the backbone is used. The top of the original MobileNetV2 is removed and the output is taken from block 12^{th}. Next, a global max pooling layer is applied to shrink the tensor into a vector. Moreover, this layer is followed by a batch normalization layer and a 0.2-dropout layer respectively. The dropout layer is added to deal with the overfitting problem. Finally, a 224-dimensional fully connected dense layer is added to be the output of this model. This tailored human feature extraction model has a total of 0.6 millions parameters. This lightweight architecture enables this model to be implemented on edge devices.

c, Distance function

With each image fed into the model, one representative feature vector is desired to be obtained. However, not every detection has a corresponding vector. Only detections with a height-over-width ratio smaller than 10 and greater than 0.9 have their embeddings. This rule was set to avoid extracting features for bounding boxes of occluded people or wrong detections because utilizing these bad detections worsens the performance of the module. The similarity between two human images is determined by comparing the corresponding feature vectors using a distance function. Specifically, the distance function is a slight modification of the cosine similarity function which is described in Equation 3.1. This distance ranges from 0 to 1. The smaller the distance is, the more similar the pair of vectors is.

$$f(A, B) = \frac{1 - g(A, B)}{2} \tag{3.1}$$

$$g(A, B) = \frac{A \cdot B}{\|A\|_2 \|B\|_2} \tag{3.2}$$

In which:

- f is the desired distance function, g is the cosine similarity function.
- The numerator in the cosine similarity function is the dot product of two

vectors A and B.

- $\|A\|_2$ is the L2 norm of vector A.
- $\|B\|_2$ is the L2 norm of vector B.

3.2.3 Human tracking

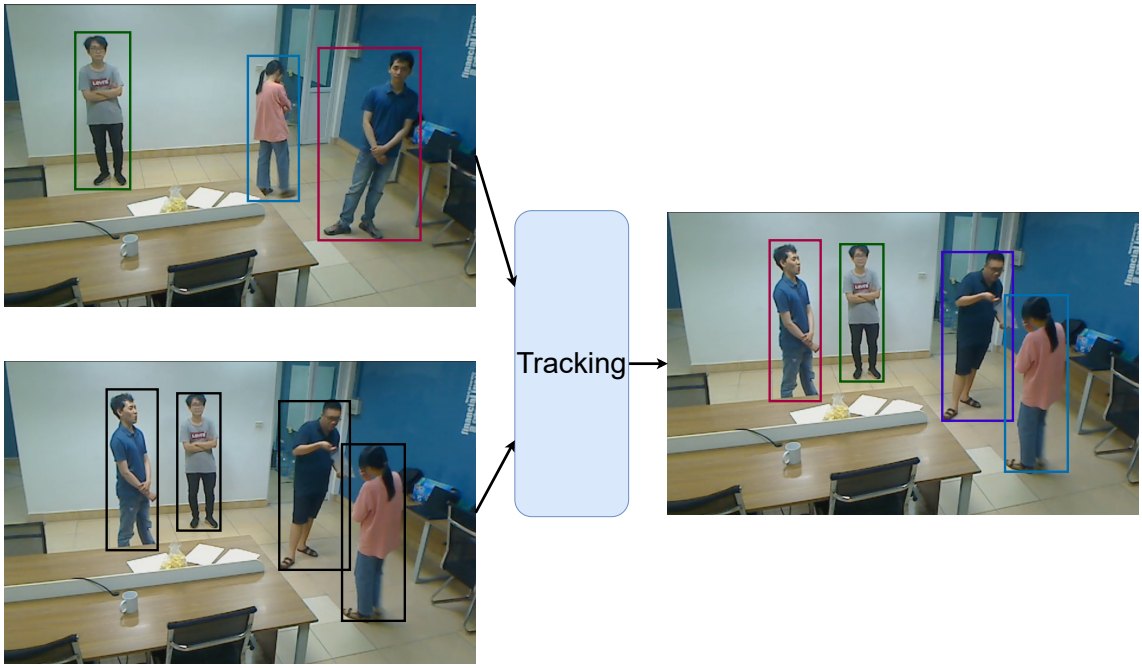

Figure 3.6: Overview of the tracking algorithm.

The human tracking algorithm helps to track individuals moving in the frames, it helps to remain the same identity for a person moving in a camera. Figure 3.6 demonstrates the function of the tracking algorithm, it remains the identity for each person moving in a camera. The tracking algorithm takes bounding boxes from the detection model in Section 3.2.1 and tracked objects from the previous frame as input. Moreover, the feature of each bounding box is extracted by the extraction model in Section 3.2.2 to additionally provide visual information for the tracking algorithm. The outputs of this algorithm are tracked entities throughout the input stream. Each entity has a unique identification number, and people from different frames with the same identification number are the same person. The information about tracked humans including the identification number, coordinates of the bounding box, and extracted feature are sent to the server. Then, this information can help the system to match an entity from one camera to another camera and collect the statistics of people entering the room.

a, Association cost function

First of all, I introduce three association cost functions which are used in the next part, Data association, where all the tracking works happen. These functions

calculate the cost between tracked objects and detected bounding boxes produced by the detection model or between two lists of tracked objects. The lower the cost is, the higher probability that the detected bounding box and the tracked object, or two tracked objects are matched. The first association cost function is described in Equation 3.3.

$$c_1(A, B) = 1 - \frac{A \cap B}{A \cup B} \tag{3.3}$$

In which:

- A and B are detected bounding boxes or estimated bounding boxes of tracked objects using the Kalman filter [33].
- $A \cap B$ is the intersection area of two bounding boxes.
- $A \cup B$ is the union area of two bounding boxes.

The second and the third association cost functions are defined as follows:

$$c_2(A, B) = f(M(A), M(B)) \tag{3.4}$$

$$c_3(A, B) = \lambda \times c_1(A, B) + (1 - \lambda) \times c_2(A, B) \tag{3.5}$$

In which:

- A and B are detected bounding boxes or estimated bounding boxes of tracked objects using the Kalman filter [33].
- c_1 is defined in Equation 3.3 as the first association cost function.
- f is defined in Equation 3.1 as the visual distance function.
- M represents the function that maps from ROI to the embedding vector by feeding the ROI into the human feature extraction model. However, in this case, if X is a tracked object, $M(X)$ refers to extracted feature vector of detection from the previous frame of the tracked object X.
- λ is a weight factor. In this case, 0.3 is chosen to be the value of λ.

b, Data association

Data association is the main component in charge of tracking objects. It helps to assign detections or newly initialized tracked objects with existing tracked objects. This algorithm takes two lists A and B as input: one list of tracked entities and one list of detections or two lists of tracked entities. Then, it creates a cost matrix

of these two lists based on an association cost function and optimally assign each bounding box or newly initialized tracked object with each tracked target by the Hungarian algorithm. Moreover, there is also a threshold to prevent assignments where detection is too different from the estimation of a tracked object.

Before going into steps to track humans, I first introduce two states of a tracked object. Each tracked objects have an important attribute called $hit\_counter$. The first state of a tracked object is called $alive$ state, it is the state where $hit\_counter$ is greater than 0. If a tracked object is matched with a bounding box, its $hit\_counter$ will increase. Moreover, $hit\_counter$ decreases when the corresponding tracked object doesn't have any match. If $hit\_counter$ is higher than a predefined threshold, that tracked object will be officially initialized. It helps the algorithm to be robust against false detections. On the other hand, if the $hit\_counter$ drops below 0, the tracked object moves to the $dead$ state. In this state, the life cycle of a tracked object does not immediately end. A tracked object will only be completely removed if it is in the $dead$ state and does not have any match in several consecutive frames. Otherwise, if a $dead$ tracked object is matched, the $hit\_counter$ will increase above 0 and this tracked object is moved back to the $alive$ state.

Next, four main steps of tracking are discussed, these four main steps are executed in every frame. In the first step, A contains initialized $alive$ tracked objects, and B contains new detections. The association cost function is c_3 in Equation 3.5 with a threshold of 0.6. In this step, if either c_1 in Equation 3.3 is greater than 0.98 or one of the bounding boxes of A or B does not have a feature embedding, then the cost will be set to 1. This step uses the cost function utilized from both geometry and visual features which may improve the tracking performance in occlusion scenarios. However, if bounding boxes are too different from each other or any bounding box does not have an embedding feature, they can not be matched.

In the second step, A contains unmatched initialized $alive$ tracked objects from step one. B consists of unmatched detections from step one. The cost function is c_1 in Equation 3.3 with a threshold of 0.7. This step only considers the coordinates of bounding boxes to calculate the cost.

Next, in the third step, A contains uninitialized $alive$ tracked objects and B contains unmatched detections from the second step. The association cost function is c_1 in Equation 3.3 with a threshold of 0.7. This step also only considers the coordinates of bounding boxes to calculate the cost.

In the final step, A is the combination of unmatched initialized $alive$ tracked objects from step two and $dead$ tracked objects. B consists of matched uninitialized

$alive$ tracked objects from step three. The cost function is c_2 in Equation 3.4 with a threshold of 0.2. This final step is used to match objects that have disappeared from the camera for a short period. In addition, it also improves the performance in occlusion scenarios.

After tracked objects are obtained, information about tracked people including the identification number, coordinates of the bounding box, and extracted feature are sent to the server. Based on received information from multiple cameras, the server can know exactly when an individual enters the room, which room he or she enters, and where he or she goes to in that specific room.

3.3 Implementation of hardware

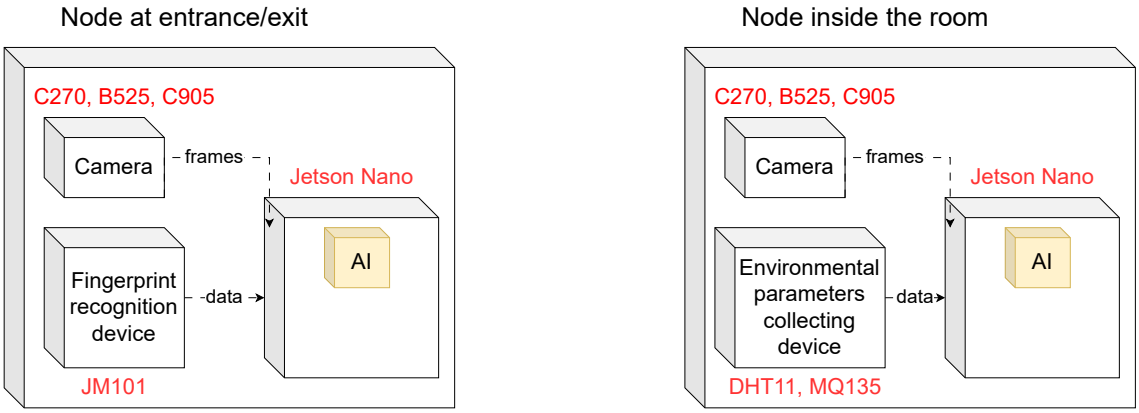

Figure 3.7: Demonstration of hardware implementation with red texts are the hardware.

This section focuses on the usage of hardware for this thesis, the reason to choose them, and their specifications including computing edge devices, Jetson Nano, webcams, and other peripheral devices. The utilization of hardware is shown in Figure 3.7.

3.3.1 NVIDIA Jetson Nano developer kit

NVIDIA Jetson Nano developer kit [41] is a small, compact single-board computer developed for AI applications. It enables applications to run multiple neural networks in parallel. It is suitable for embedded systems, robots that require high-performance inference. Jetson Nano was chosen instead of Raspberry Pi 4 [42] because it has stronger computing power. An edge device with too low computing power does not have the capability to run many deep learning models on a single board even when models have been tailored to be lighter. The detailed technical specifications of Jetson Nano can be seen in Table 3.1. Jetson Nano can run with a minimum power consumption of only 5 watts and a maximum of up to 10 watts.

CPU	Quad-core ARM A57 @ 1.43 GHz
GPU	128-core Maxwell
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
USB	4x USB 3.0, USB 2.0 Micro-B
Display	HDMI and display port

Table 3.1: Technical specifications of NVIDIA Jetson Nano Developer Kit.

3.3.2 Webcams

This thesis uses three types of USB webcams: Logitech C270 HD, Logitech HD B525, and Logitech HD C905. They are connected to the Jetson Nano via a USB port. All of them have a video resolution of 720p. Each Jetson Nano has one webcam.

3.3.3 Other devices

Jetson Nano is also connected to other peripherals such as the JM101 fingerprint sensor, environment sensors like the DHT11 temperature and humidity sensor, and the MQ135 air quality sensor. However, the development of applications using these devices is carried out by other members of the research team and is beyond the scope of this thesis. These devices can be seen in Figure 3.8 and Figure 3.9.

3.4 Utilization of frameworks

This section focuses on the utilization of frameworks for this thesis which is shown in Figure 3.10.

3.4.1 Tensorflow and Pytorch

TensorFlow [43] and PyTorch [44] are used to develop the human detection model and human feature extraction model of this thesis. Both of the models are later converted to TensorRT format for efficient inference on Jetson Nano. They are two of the most widely used open-source machine learning, deep learning frameworks. They are used by a large number of engineers, scientists all over the world to develop AI models and applications.

3.4.2 TensorRT

TensorRT [45] is a library designed for high-performance deep learning inference that contains an optimizer and runtime for deep learning inference. It enables AI applications to achieve low latency and high throughput. TensorRT can optimize models to run especially well on NVIDIA GPUs. Therefore, TensorRT is used to optimize the human detection model and the human feature extraction model of this thesis to run on Jetson Nano. TensorRT uses post-quantization to increase the inference speed of the human detection model by converting its weights from 32-bit

Figure 3.8: Fingerprint sensor JM101.

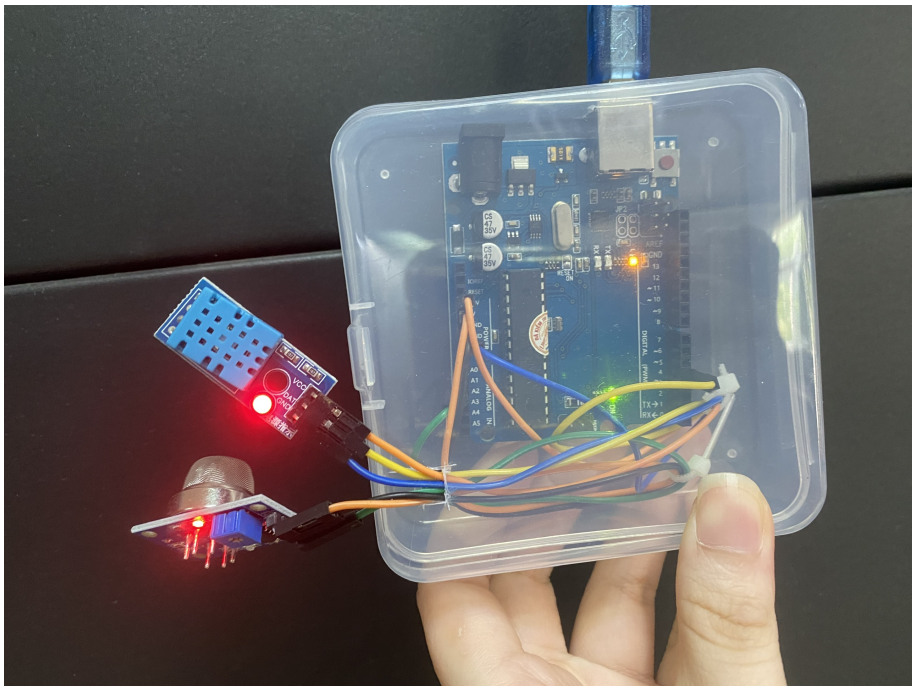

Figure 3.9: Temperature, humidity sensor DHT11 (blue rectangle) and air quality sensor MQ135 (round head).

to 16-bit floating points. On the other hand, the feature extraction model is already

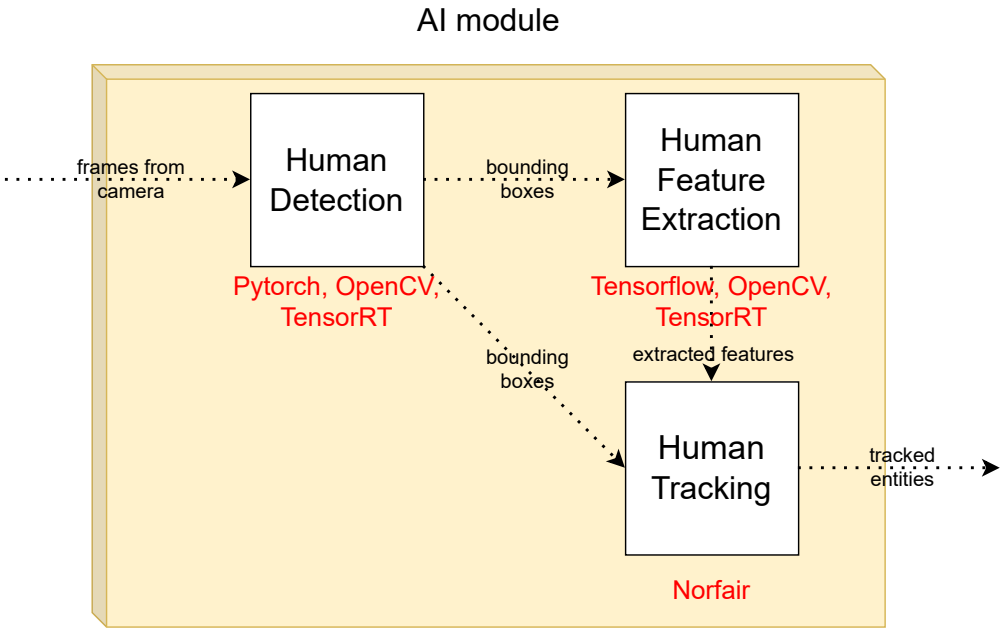

Figure 3.10: Demonstration of frameworks utilization with red texts are the frameworks.

lightweight, and therefore its weights are kept as 32-bit floating points to maintain the performance of the model.

3.4.3 OpenCV

OpenCV [46] is an open-source image processing and computer vision library. OpenCV was chosen to be used because it provides easy-to-use APIs to read frames from connected USB cameras. Moreover, OpenCV also provides many useful image-processing APIs. It was used in the pre-processing and post-processing parts of the human detection and human feature extraction model.

3.4.4 Norfair

Norfair [47] is a customizable lightweight Python library for real-time multi-object tracking. Norfair brings convenience when you want to modify a tracking algorithm according to your preferences. Norfair makes it extremely convenient to integrate your own detection model or cost function into the tracking algorithm. Norfair was used to develop the tracking algorithm in this thesis. It helps to integrate the human detection model, the feature extraction model, and the customized cost function into the tracking algorithm easily.

This chapter has presented the flow of the AI module and details about each component in this module. Each component has been carefully researched and customized to solve the raised problem and achieve the objectives of this thesis. Evaluations and deployment results are discussed in the next chapter, Chapter 4.

CHAPTER 4. EXPERIMENTAL RESULTS

4.1 Environment setup

For evaluation results, all the training processes are conducted on a computer that has an Intel Core i3-9100F CPU and a GTX1660 GPU. However, all the tests are only executed on the Nvidia Jetson Nano Developer Kit which has the technical specifications described in Table 3.1.

For deployment, each edge node will be a combination of a Jetson Nano, a USB camera, and other peripheral devices. The deployed human feature extraction model was obtained by training with the PRW dataset [48] and the CUHK03 dataset [49] as described in Section 4.2.

4.2 Datasets

For deployment and evaluation purposes, two public datasets: PRW dataset [48] and CUHK03 dataset [49] were utilized. Images from these two datasets can be seen in Figure 4.1 with two images on the left from the PRW dataset and two images on the right from the CUHK03 dataset.

Figure 4.1: Images from PRW and CUHK03 datasets.

PRW dataset [48] was collected using six cameras at Tsinghua University during the summer of 2014. There are a total of 11,816 labeled video frames with 932 identities and 34,304 bounding boxes. This dataset is designed for cross-camera retrieval.

CUHK03 dataset [49] consists of 14,097 pictures of 1,467 individuals, collected using six cameras on campus. Two cameras were used for each individual. There are two types of bounding box annotations in this dataset: manually labeled and automatically labeled using a detector. The dataset contains 20 train/test partitions.

Our approach only uses the manually labeled portion of the dataset for training, evaluation, and comparison purposes in any experiments or deployment.

For evaluation, the feature extraction model was trained and tested only on the CUHK03 dataset because of its popularity. The model was trained for every partition and the final result was averaged from results of 20 partitions. In each partition, 100 identities are separated for testing, 100 identities are split for validation, and the rest are in the training set. The training settings and evaluation results can be seen in Section 4.5.

For deployment, I combined the data from both of the datasets into one big dataset and use this dataset to train the model. I took out 3,439 images of 100 identities in the PRW dataset for validating purposes and these 100 identities would not appear in the training set. Hyper-parameters and strategy for training are identical to those specified in Section 4.5.1.

4.3 Quantitative metrics

4.3.1 IoU

IoU stands for Intersection over Union. If there are two bounding boxes, the IoU score of these two bounding boxes is calculated by the intersection of two boxes over the union of them which is shown in Equation 4.1.

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \tag{4.1}$$

In which:

- A and B are bounding boxes.
- $A \cap B$ is the intersection area of two bounding boxes.
- $A \cup B$ is the union area of two bounding boxes.

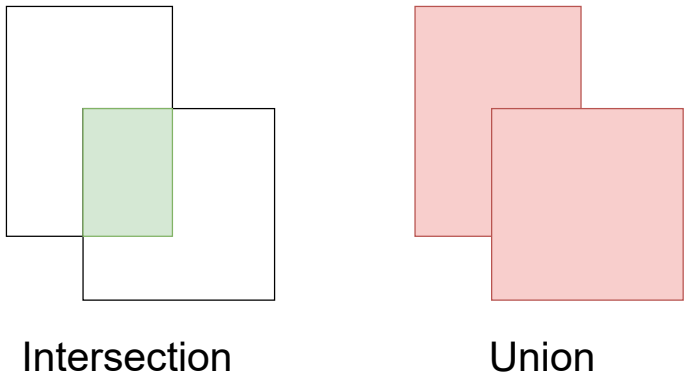

Figure 4.2: Demonstrations of intersection and union in IoU.

Intersection and union are demonstrated in Figure 4.2.

4.3.2 EER

EER stands for Equal Error Rate. EER is a performance metric used to evaluate the accuracy of ReID algorithms in matching pairs of images. EER is the point where the false rejection rate (FRR) and false acceptance rate (FAR) are equal. For example, in a test set, pair-wise distance is calculated for every possible pair of images. Then, the threshold will be adjusted so that the FAR and FRR are equal. At that point, they are also equal to the EER score.

4.3.3 Re-ID mAP

mAP is used in Re-ID algorithms as a performance metric to evaluate the accuracy. It is a measure of how well an algorithm can rank a set of images in order of similarity to a query image.

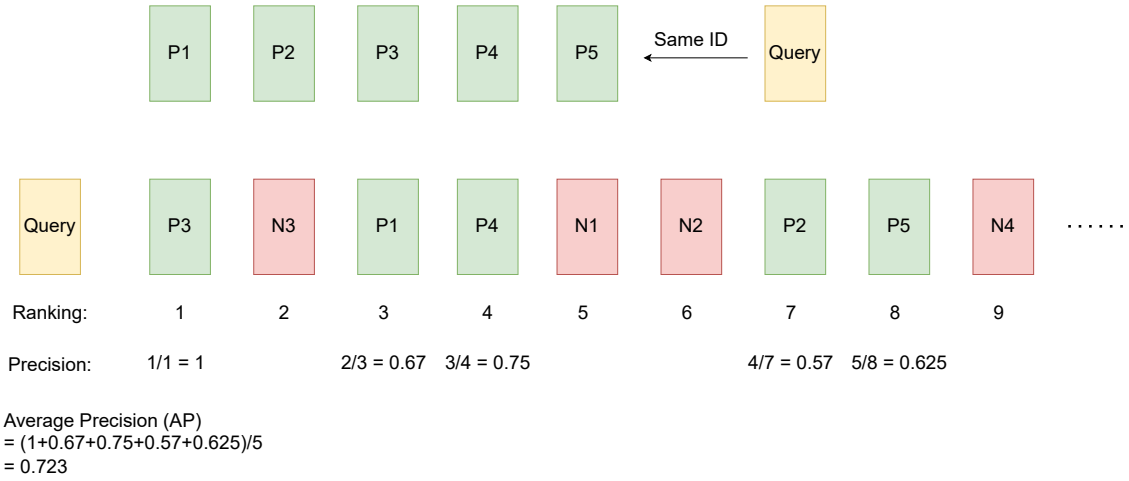

Figure 4.3: An example for average precision calculation for a query image. The yellow box is the query sample. Green boxes are samples with the same identity as the query image and red boxes are samples with different identities from the query image.

Figure 4.3 illustrates the average precision calculation process for a query image. P1 to P5 have the same identity with the query image and N1 to N4 have different identities from the query image. Next, the mean of average precision of all samples in the test set is computed to obtain the final mAP score.

4.3.4 Rank-1

Rank-1 is a performance metric used to evaluate the accuracy of Re-ID algorithms. Rank-1 accuracy measures the percentage of cases where the top-ranked match returned by the algorithm is correct.

Rank-1 accuracy is calculated by comparing the identity of the ground truth query image with the identity of the top-ranked match. Distances between the ground truth query image and all images in the gallery are calculated. Then, the results are sorted by descending the similarity. If the most similar sample in the

gallery has the same identity as the query image, this is considered a correct prediction. The Rank-1 accuracy is the percentage of cases where the algorithm makes a correct prediction for the top-ranked match.

4.4 Experimental results of detection model

Table 4.1 shows the evaluation results of detection models on Jetson Nano. Three versions of the model were evaluated. The model is either in the native Pytorch format or in the TensorRT format. TensorRT-converted models were quantized to 16-bit floating points.

Model	Input size	Converted to TensorRT	Inference time (milliseconds)
YOLOv5n	640×640	No	47.4
YOLOv5n	640×640	Yes	44.1
YOLOv5n (used in deployment)	384×384	Yes	17.9

Table 4.1: Results of detection models on Jetson Nano.

As seen in Table 4.1, converting the model to TensorRT format has improved its inference time from 47.4 milliseconds to 44.1 milliseconds. Furthermore, if the input size is set to 384×384, the inference time of the converted model significantly reduces to 17.9 milliseconds.

4.5 Experimental results of feature extraction model

4.5.1 Hyper-parameters and training settings

Table 4.2 provides a summary of the hyperparameters used in this model. The input was pre-processed to have the size 192×64 as described in Section 3.2.2.a. A batch size of 64 was selected. During the training step, one person from the dataset was chosen, and all samples of this person were added to the mini-batch. The remaining samples in the batch were randomly selected from other people. An epoch ended when all the people in the training set were iterated through. Hard triplet loss with the margin constant of 0.2 was used. Adam [50] was utilized as the optimizer with the initial learning rate of 1×10^{-4} The model was trained for 500 epochs, and the final model was chosen based on the lowest EER score on the validation set.

Input size	Batch size	Margin	Initial learning rate	Number of epochs
192×64	64	0.2	1×10^{-4}	500

Table 4.2: Hyperparameters for human feature extraction model.

4.5.2 Evaluation

Table 4.3 shows the experimental results on the CUHK03 dataset. Two versions of the proposed feature extraction model were evaluated. The first model is a native Tensorflow model and the second one is the model which has been converted to TensorRT engine.

Model	Re-ID mAP (%)	Rank-1 (%)	Parameters (millions)	Inference time (milliseconds)
PCB [51]	54.2	61.3	26.8	-
LightMBN [52]	85.1	87.2	9.3	-
Pyramid [53]	76.9	78.9	29.1	-
DSA-reID [54]	75.2	78.9	38.5	-
MPN [55]	81.1	85.0	31.5	-
MGN [56]	67.4	68.0	68.8	-
Customized MobileNetV2 (Before conversion)	75.7	79.1	0.6	23.4
Customized MobileNetV2 (After conversion)	75.7	79.1	0.6	4.1

Table 4.3: Results of feature extraction models on the CUHK03 dataset before and after conversion.

As seen in Table 4.3, while the accuracy of the proposed model is not on par with state-of-the-art models, it is significantly lighter than all other models. In fact, the second-lightest model has about 15 times as many parameters as the proposed model. Despite this, the proposed model still achieves a decent level of accuracy. In addition, without quantization, the precision of both versions of the proposed model is equal. However, the converted model is nearly five times faster than the original model. Moreover, the inference time of the proposed feature extraction model with the original MobileNetV2 backbone is 6.4 milliseconds. This model is 56% slower than the customized model. When there is only one person in the frame, the difference is negligible. However, if the number of people in the frame increases, without customization, MobileNetV2 will make the module extremely slower.

4.6 Deployment results

4.6.1 Setup

The full system is deployed in room 405 at the B1 building at Hanoi University of Science and Technology. There are a total of three Jetson Nano devices to be deployed: one device at the entrance to the room, one device in the large room, and one device in the small room. Figure 4.4 and Figure 4.5 show two rooms where monitoring edge devices were deployed. Figure 4.6 shows the deployment of the

entrance node. Figure 4.7 and Figure 4.8 illustrate the deployment of the room node inside the large room and the small room.

Figure 4.4: The small room.

In each node inside the room, a USB camera and other peripheral devices will be connected to a Jetson Nano kit like in Figure 4.9.

4.6.2 Operating speed experiments

For further testing purposes, the speed test was conducted on Jetson Nano with the number of people inside the room increased. The overall testing frame rate of Jetson Nano is shown in Table 4.4. A single Jetson Nano conducted all the work including running two deep learning models and one tracking algorithm, receiving environment parameters from sensors and sending them with tracked human information each second, and sending raw input frames to the server every two seconds.

When there were no more than two individuals present in the image, Jetson Nano was capable of achieving a frame rate greater than 20 FPS. However, as the number of individuals captured in the image increased, the frame rate also decreased. Specifically, starting from seven people, the FPS fell below 10. A significantly low frame rate can degrade the performance of the tracking algorithm [57]. Therefore,

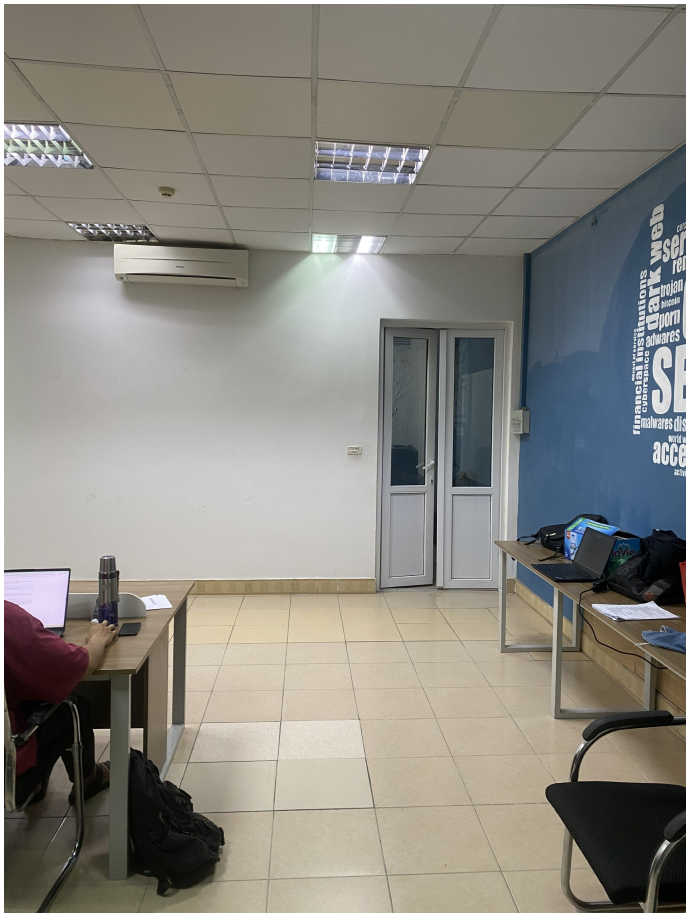

Figure 4.5: The large room.

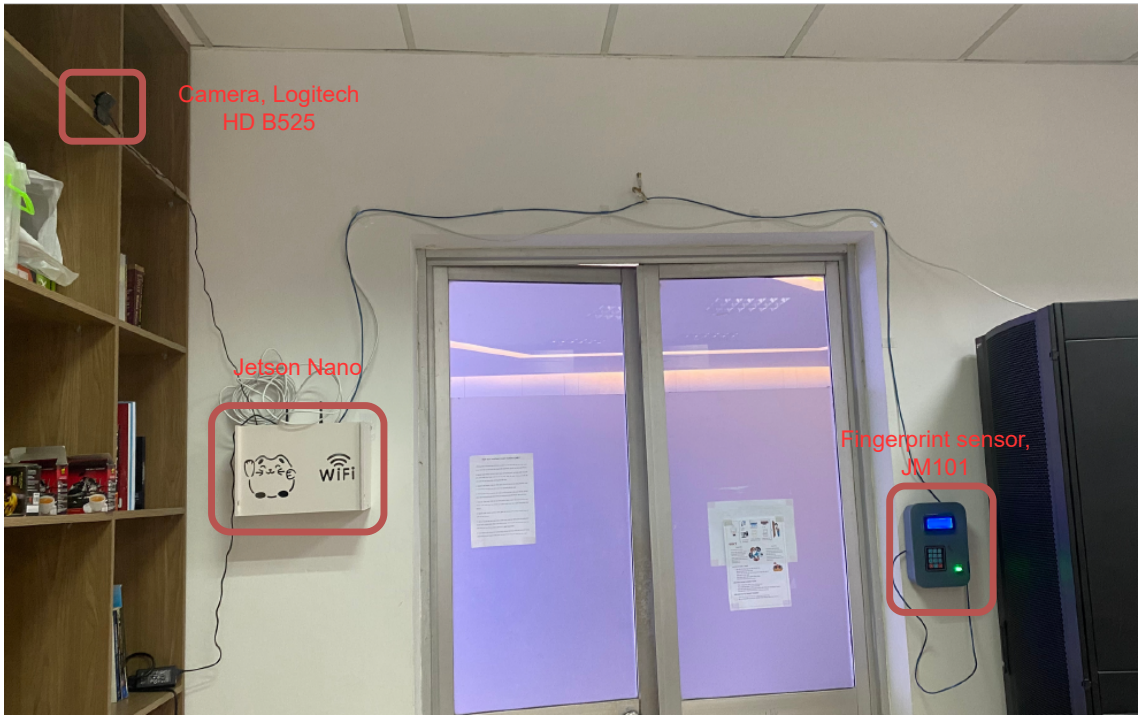

Figure 4.6: Deployment of entrance node.

for crowded scenarios, stronger hardware needs to be used or models and algorithms need to be further improved.

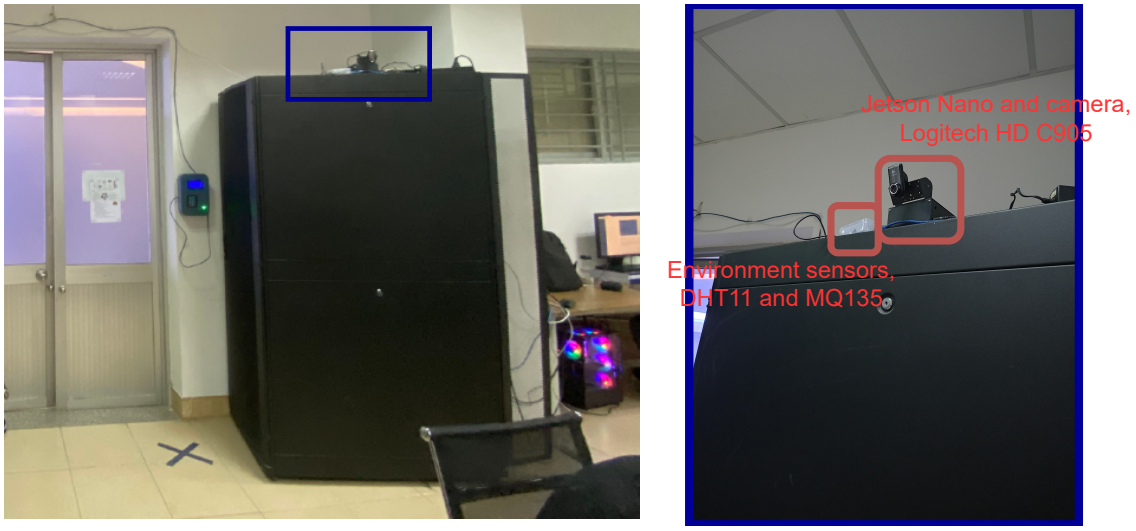

Figure 4.7: Deployment of room node inside the large room.

Figure 4.8: Deployment of room node inside the small room.

Number of people	FPS
0	34.4
1	25.1
2	20.9
3	17.1
4	15.1
5	13.0
6	11.2
7	9.6

Table 4.4: FPS test results when increasing the number of people in a single camera frame.

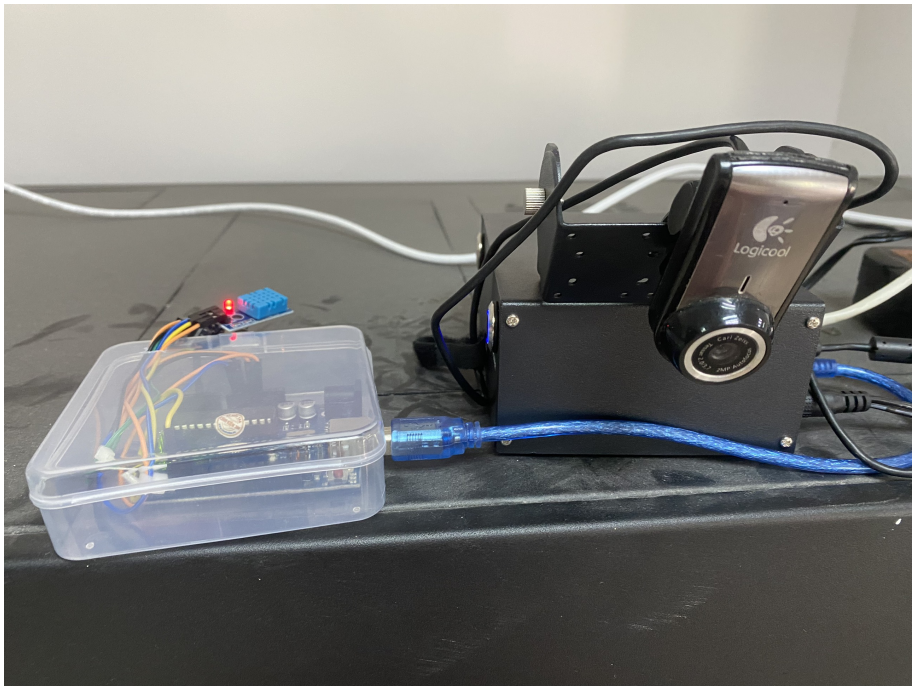

Figure 4.9: Jetson Nano with a camera and environment sensors.

4.6.3 Functionalities evaluation

The functionalities of the AI module deployed on edge devices were evaluated through the output at the end application. The functionalities are shown by pictures of the user interface at the end application. Figure 4.10 is a showcase of the interface of the application.

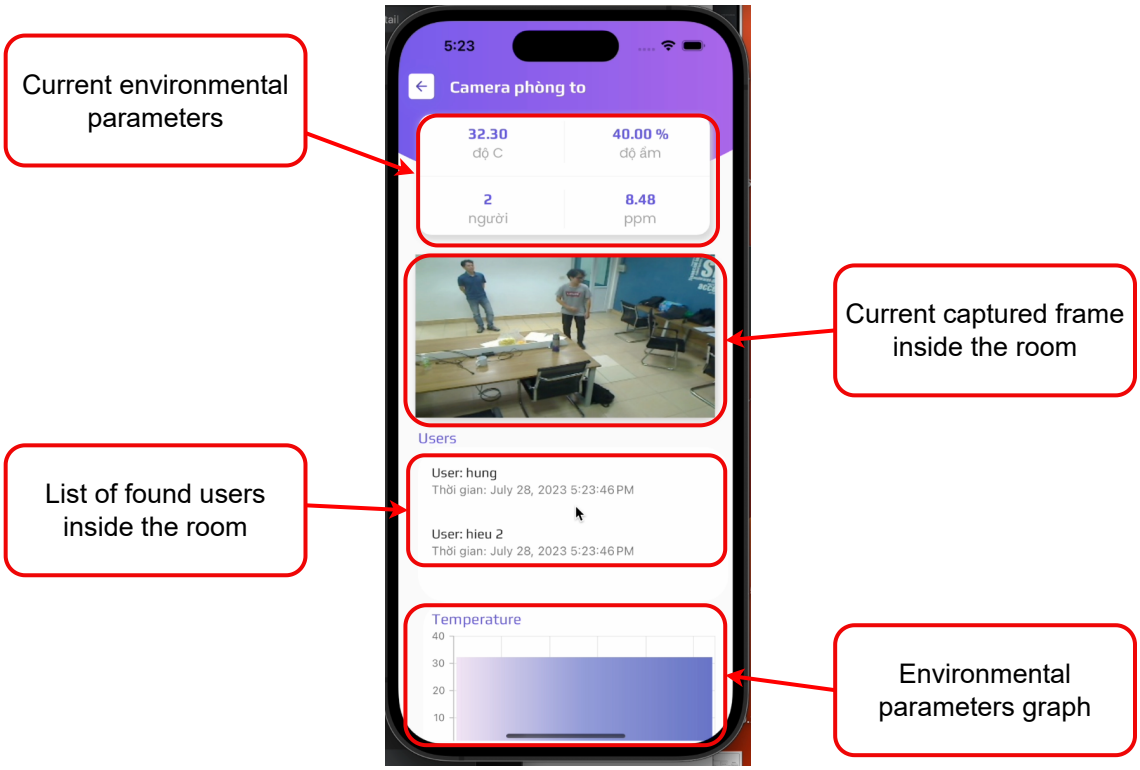

Figure 4.10: The interface of the application.

a, Take attendance and identification when entering the room

When a person checks attendance at the entrance by his or her fingerprint, the system will start to track that person inside the room. Figure 4.11 shows pictures of attendance checking and the result when a person enters the room. Jetson Nano can identify that person and send the information to the server.

Figure 4.11: Attendance checking at entrance and interface of the application when there is a person entering the room.

b, Re-ID when moving to another room

The system re-identified people when they switched from the large room to the small room as shown in Figure 4.12. All information including when a person entered the room, how they moved in the room, and which room they visited are all collected. This information was sent from multiple Jetson Nano and processed by the server.

c, Re-ID when disappearing from the frame for a period

Sometimes, an object may obstruct a person from the camera's view. As an illustration in Figure 4.13, the person bent down to retrieve a pencil under a table and becomes obscured. When he reappeared after a short time, he was re-identified by the system. If a person disappears from all cameras for a long period of time, that person will be considered leaving the room.

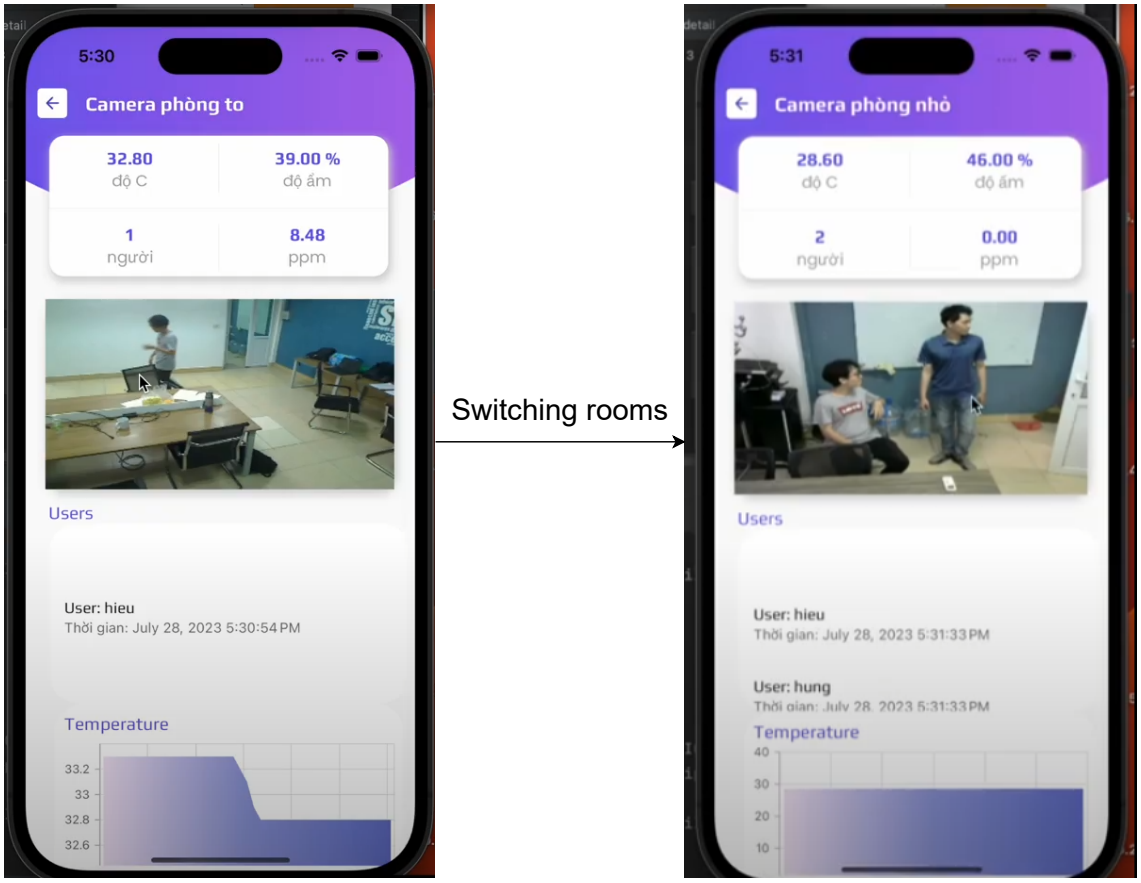

Figure 4.12: Results when switching rooms.

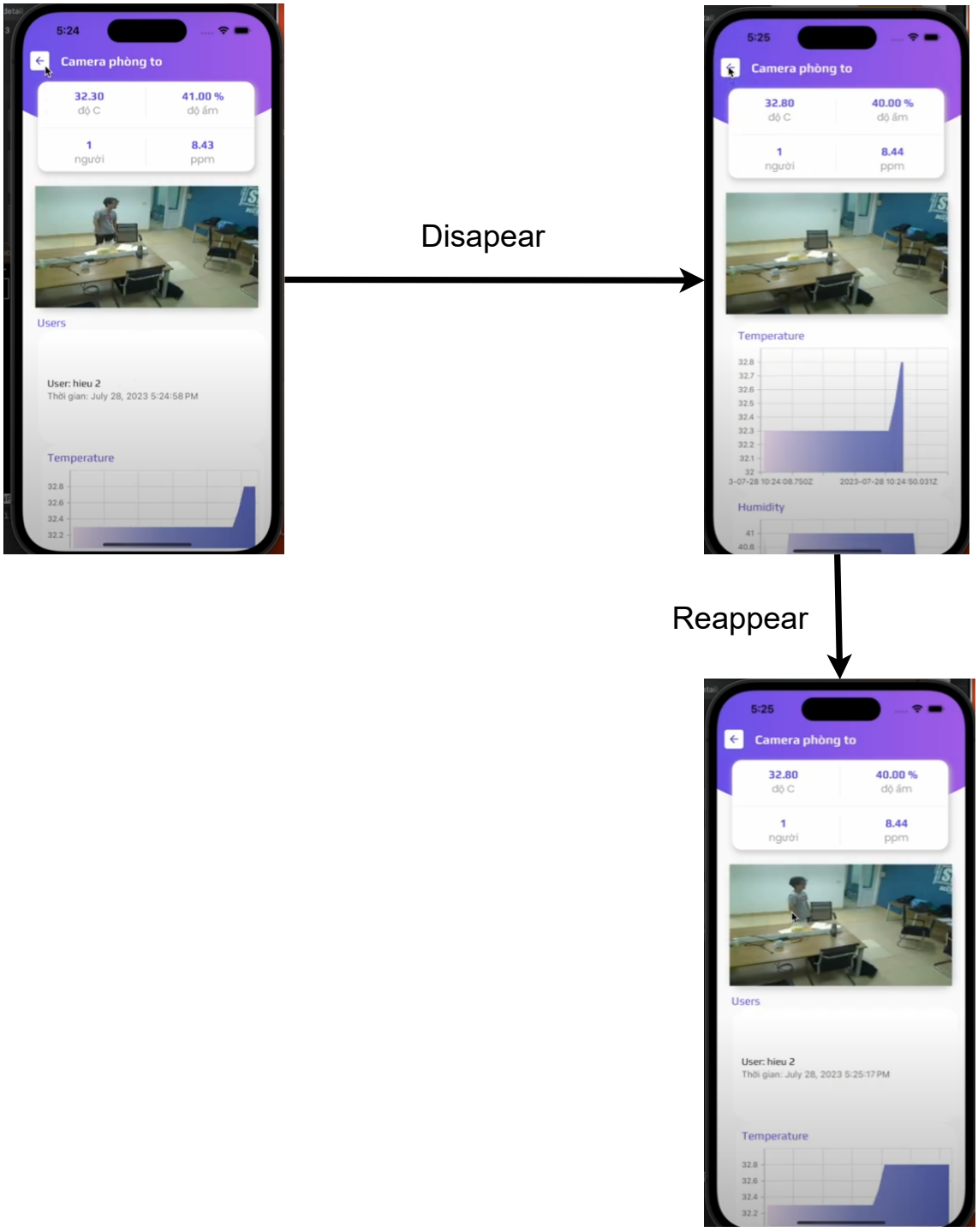

Figure 4.13: Results when a person disappears from the frame from a period and reappears.

CHAPTER 5. CONCLUSIONS AND FUTURE WORKS

5.1 Achievements

The thesis has presented related research, foundation theories, and frameworks required to create an AI module for the person monitoring on edge devices using computer vision. All of these components have been combined to create a working prototype that meets the original objectives:

1. Developed a stand-alone remote human monitoring AI module, implemented on edge devices, Jetson Nano. The module contains three main parts: human detection, human feature extraction, and human tracking. The proposed module could detect humans, track them, and give each individual identity features based on their outlook. The output of this module was integrated into a bigger management system.
2. Implemented deep learning models directly on edge devices with input source taken from a connected USB camera. The object detection and feature extraction models were tailored and optimized for deployment on Jetson Nano. The frame rate was higher than ten if the number of people was smaller or equal to six. A prototype of the proposed AI module has been put into practical use and examined in room 405, B1 building at Hanoi University of Science and Technology.

Furthermore, while completing this thesis, I have gained a lot of useful knowledge about computer vision, AI algorithms, and their implementation on edge devices. I find deploying them on real-world devices really challenging. All of these have become a precious experience for me.

5.2 Limitations

Because of the limited time, money, and computing resource constraints, this thesis faces the following limitations:

1. Jetson Nano does not function well when the number of people in the frame is high.
2. The viewing angle of the camera does not cover the whole room and the system will not work well when there are many objects in the room that obscure the view of the camera.
3. The proposed feature extraction model can meet the lightweight requirements to be implemented on edge devices but it has to trade off with accuracy.

4. In the event that the edge device experiences a shutdown, there is no automated recovery protocol.

5.3 Suggestions for future works

To provide better performance, several future developments can be conducted:

1. Using more powerful hardware thereby not only directly increases FPS but also creates a good condition to implement more accurate algorithms.
2. Utilizing a camera with a broader field of view and installing it on the ceiling to achieve comprehensive room monitoring while avoiding blind spots.
3. Lightweight, high-accuracy models and algorithms need to be further researched.
4. Installing an automated recovery protocol to turn on edge devices.
5. Carrying out more real-world deployments to spot weaknesses that need improvement.

REFERENCE

[1] **url**: https://thoibaotaichinhvietnam.vn/doanh-nghiep-viet-nam-da-u-a-n-tang-truo-ng-qua-ke-t-qua-to-ng-die-u-tra-kinh-te-nam-2021-109675.html (**urlseen** 27/06/2023).

[2] **url**: https://www.globenewswire.com/news-release/2021/06/22/2251205/0/en/Time-Tracking-Software-Market-to-Exhibit-20-CAGR-by-2026-Report-by-Market-Research-Future-MRFR.html (**urlseen** 10/07/2023).

[3] O. Elharrouss, N. Almaadeed **and** S. Al-Maadeed, "A review of video surveillance systems," *Journal of Visual Communication and Image Representation*, **journvol** 77, **page** 103 116, 2021.

[4] J. Neves, F. Narducci, S. Barra **and** H. Proença, "Biometric recognition in surveillance scenarios: A survey," *Artificial Intelligence Review*, **journvol** 46, **pages** 515–541, 2016.

[5] **url**: https://base.vn/platform/hrm (**urlseen** 27/06/2023).

[6] **url**: https://amis.misa.vn/amis-nhan-su/ (**urlseen** 27/06/2023).

[7] M. Sarwar Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan **and** F. Hussain, "Machine learning at the network edge: A survey," *arXiv e-prints*, arXiv–1908, 2019.

[8] X. Zhang, X. Zhou, M. Lin **and** J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2018, **pages** 6848–6856.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov **and** L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2018, **pages** 4510–4520.

[10] S. Karanam, Y. Li **and** R. J. Radke, "Person re-identification with discriminatively trained viewpoint invariant dictionaries," **in***Proceedings of the IEEE international conference on computer vision* 2015, **pages** 4516–4524.

[11] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao **and** S. C. Hoi, "Deep learning for person re-identification: A survey and outlook," *IEEE transactions on pattern analysis and machine intelligence*, **journvol** 44, **number** 6, **pages** 2872–2893, 2021.

[12] T. Matsukawa, T. Okabe, E. Suzuki **and** Y. Sato, "Hierarchical gaussian descriptor for person re-identification," **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2016, **pages** 1363–1372.

[13] M. Farenzena, L. Bazzani, A. Perina, V. Murino **and** M. Cristani, "Person re-identification by symmetry-driven accumulation of local features," **in***2010 IEEE computer society conference on computer vision and pattern recognition* IEEE, 2010, **pages** 2360–2367.

[14] A. Taivalsaari **and** T. Mikkonen, "On the development of iot systems," **in***2018 Third international conference on fog and mobile edge computing (FMEC)* IEEE, 2018, **pages** 13–19.

[15] N. S. Sworna, A. M. Islam, S. Shatabda **and** S. Islam, "Towards development of iot-ml driven healthcare systems: A survey," *Journal of Network and Computer Applications*, **jourvol** 196, **page** 103 244, 2021.

[16] K. Lakhwani, H. Gianey, N. Agarwal **and** S. Gupta, "Development of iot for smart agriculture a review," **in***Emerging Trends in Expert Applications and Security: Proceedings of ICETEAS 2018* Springer, 2019, **pages** 425–432.

[17] R. Singh **and** S. S. Gill, "Edge ai: A survey," *Internet of Things and Cyber-Physical Systems*, 2023.

[18] P. Agarwal **and** M. Alam, "A lightweight deep learning model for human activity recognition on edge devices," *Procedia Computer Science*, **jourvol** 167, **pages** 2364–2373, 2020.

[19] Y. Pang, Y. Yuan, X. Li **and** J. Pan, "Efficient hog human detection," *Signal processing*, **jourvol** 91, **number** 4, **pages** 773–781, 2011.

[20] R. Girshick, J. Donahue, T. Darrell **and** J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2014, **pages** 580–587.

[21] K. O'Shea **and** R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[22] R. Girshick, "Fast r-cnn," **in***Proceedings of the IEEE international conference on computer vision* 2015, **pages** 1440–1448.

[23] S. Ren, K. He, R. Girshick **and** J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, **jourvol** 28, 2015.

[24] J. Redmon, S. Divvala, R. Girshick **and** A. Farhadi, "You only look once: Unified, real-time object detection," **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2016, **pages** 779–788.

[25] G. Jocher, A. Chaurasia, A. Stoken **andothers**, *ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation*, **version** v7.0, **november**

2022. DOI: 10.5281/zenodo.7347926. **url**: https://doi.org/10.5281/zenodo.7347926.

[26] D. Reid, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, **jourvol** 24, **number** 6, **pages** 843–854, 1979.

[27] A. Bewley, Z. Ge, L. Ott, F. Ramos **and** B. Upcroft, "Simple online and realtime tracking," **in***2016 IEEE international conference on image processing (ICIP)* IEEE, 2016, **pages** 3464–3468.

[28] N. Wojke, A. Bewley **and** D. Paulus, "Simple online and realtime tracking with a deep association metric," **in***2017 IEEE international conference on image processing (ICIP)* IEEE, 2017, **pages** 3645–3649.

[29] L. Mekkayil **and** H. Ramasangu, "Object tracking with correlation filters using selective single background patch," *arXiv preprint arXiv:1805.03453*, 2018.

[30] D. S. Bolme, J. R. Beveridge, B. A. Draper **and** Y. M. Lui, "Visual object tracking using adaptive correlation filters," **in***2010 IEEE computer society conference on computer vision and pattern recognition* IEEE, 2010, **pages** 2544–2550.

[31] J Vergés-Llahí, J Aranda, A Sanfeliu **andothers**, "Object tracking system using colour histograms," **in***Proceedings of the 9th Spanish Symposium on Pattern Recognition and Image Analysis* 2001, **pages** 225–230.

[32] Z. Zivkovic **and** B. Krose, "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.* IEEE, **volume** 1, 2004, **pages** I–I.

[33] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, **jourvol** 82, **number** Series D, **pages** 35–45, 1960.

[34] R. Hadsell, S. Chopra **and** Y. LeCun, "Dimensionality reduction by learning an invariant mapping," **in***2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)* IEEE, **volume** 2, 2006, **pages** 1735–1742.

[35] F. Schroff, D. Kalenichenko **and** J. Philbin, "Facenet: A unified embedding for face recognition and clustering," **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2015, **pages** 815–823.

[36] B. Deng, S. Jia **and** D. Shi, "Deep metric learning-based feature embedding for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, **jourvol** 58, **number** 2, **pages** 1422–1435, 2019.

[37] X. Li, H. Yin, K. Zhou **and** X. Zhou, "Semi-supervised clustering with deep metric learning and graph embedding," *World Wide Web*, **jourvol** 23, **pages** 781–798, 2020.

[38] R. Cao, Q. Zhang, J. Zhu **andothers**, "Enhancing remote sensing image retrieval using a triplet deep metric learning network," *International Journal of Remote Sensing*, **jourvol** 41, **number** 2, **pages** 740–751, 2020.

[39] A. Hermans, L. Beyer **and** B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[40] T.-Y. Lin, M. Maire, S. Belongie **andothers**, "Microsoft coco: Common objects in context," **in***Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13* Springer, 2014, **pages** 740–755.

[41] **url**: `https://developer.nvidia.com/embedded/jetson-nano-developer-kit` (**urlseen** 13/07/2023).

[42] **url**: `https://www.raspberrypi.com/products/raspberry-pi-4-model-b/` (**urlseen** 27/06/2023).

[43] **url**: `https://www.tensorflow.org/` (**urlseen** 13/07/2023).

[44] **url**: `https://pytorch.org/` (**urlseen** 13/07/2023).

[45] **url**: `https://developer.nvidia.com/tensorrt#:~:text=TensorRT%2C%20built%20on%20the%20NVIDIA,and%20others%20on%20NVIDIA%20GPUs.` (**urlseen** 13/07/2023).

[46] **url**: `https://opencv.org/` (**urlseen** 13/07/2023).

[47] J. Alori, A. Descoins, javier **andothers**, *Tryolabs/norfair: V2.2.0*, **version** v2.2.0, **january** 2023. DOI: `10.5281/zenodo.7504727`. **url**: `https://doi.org/10.5281/zenodo.7504727`.

[48] L. Zheng, H. Zhang, S. Sun, M. Chandraker **and** Q. Tian, "Person re-identification in the wild," *arXiv preprint arXiv:1604.02531*, 2016.

[49] W. Li, R. Zhao, T. Xiao **and** X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," **in***CVPR* 2014.

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

[51] Y. Sun, L. Zheng, Y. Yang, Q. Tian **and** S. Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," **in***Proceedings of the European conference on computer vision (ECCV)* 2018, **pages** 480–496.

[52] F. Herzog, X. Ji, T. Teepe, S. Hörmann, J. Gilg **and** G. Rigoll, "Lightweight multi-branch network for person re-identification," **in***2021 IEEE International Conference on Image Processing (ICIP)* IEEE, 2021, **pages** 1129–1133.

[53] F. Zheng, C. Deng, X. Sun **andothers**, "Pyramidal person re-identification via multi-loss dynamic training," **in***Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* 2019, **pages** 8514–8522.

[54] Z. Zhang, C. Lan, W. Zeng **and** Z. Chen, "Densely semantically aligned person re-identification," **in***Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* 2019, **pages** 667–676.

[55] C. Ding, K. Wang, P. Wang **and** D. Tao, "Multi-task learning with coarse priors for robust part-aware person re-identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **jourvol** 44, **number** 3, **pages** 1474–1488, 2020.

[56] G. Wang, Y. Yuan, X. Chen, J. Li **and** X. Zhou, "Learning discriminative features with multiple granularities for person re-identification," **in***Proceedings of the 26th ACM international conference on Multimedia* 2018, **pages** 274–282.

[57] A. Y. B. Mabrouk, G. Facciolo, R. G. von Gioi **and** A. Davy, "An assessment of multi object tracking on low framerate conditions," 2022.