DISSERTATION

DISTRIBUTED VIDEO TRACKING

by

SHARRANTH RAGU
16002514

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Computer)

JANUARY 2022

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Distributed Video Tracking

by

SHARRANTH RAGU

16002514

A project dissertation submitted to the

Computer Engineering Programme

Universiti Teknologi PETRONAS

in partial fulfilment of the requirements for the

BACHELOR OF ENGINEERING (Hons)

(COMPUTER)

Approved by,

_____

(Dr. Patrick Sebastian)

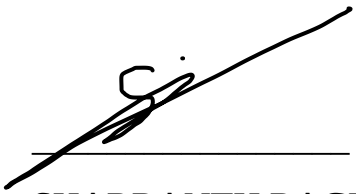
UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

January 2022

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



SHARRANTH RAGU

ACKNOWLEDGEMENT

For the past six months, I have been working on my final year project and dissertation, titled Distributed Video Tracking. Throughout the entire journey in completing this relatively challenging project, I have received immense support and assistance. With that being said, I would first take this opportunity to thank and express my extreme gratitude towards Dr. Patrick Sebastian, my final year project supervisor. My supervisor largely assisted in facilitating the overall project, providing ideas and solutions along the way.

Besides, I would like to appreciate Universiti Teknologi PETRONAS in providing such pragmatic and well organised syllabus, which made the progress of the whole project a smooth sail from day one. I believe that the final year project allows students to work on various skills, whilst honing technical understanding that may be taught in the years of pursuing degree, or knowledge that requires completely new self-learning.

I would want to specially thank my friends, that may have directly or indirectly helped me in this project. My parents and family were also my pillar of support throughout the project, both financially and emotionally.

ABSTRACT

Distributed video tracking, or widely known as *Person Re-identification* is a process or the ability of a system to associate images or frames from videos of the same person, which could be sourced from both pre-recorded and real-time footages. The developed system should be able to identify and retrieve the person of interest by combining images from various non-overlapping cameras. The purpose of Re-ID is to specifically determine whether the person of interest has appeared in another location at a different time or by the same camera at a different instant. Hence, the approach in solving the issue is to first identify distinct characteristics that define a particular person, perhaps by feeding the *training* model with a dataset of cropped images of the non-stationary subject as there is movement across the camera [2]. The project can be divided into two main algorithms; the first algorithm is mainly catered for Pedestrian detection and tracking whilst the second focuses in reidentifying the pedestrian. The project would utilize OpenCV for capturing devices, TensorFlow object detection API *(Faster-RCNN)* for person detection *(forming bounding boxes)* and *SORT* (*Simple Online and Realtime Tracking)* for tracking the person during a video input . With prebuilt OpenCV functions, cropped images would be cropped from identifying the coordinates of bounding boxes, and saved in a local database. The stored images would then be the input material to be fed into the *Siamese Neural Network* developed through TensorFlow Keras. The Siamese Model was trained individually on three different camera inputs, where two of them were from pre-recorded videos taken from a phone camera, whilst one was taken from an internet protocol camera. With an input from the first angle, an individual would linger around for an unfixed amount of time. This is where the trained images would be collected, before being fed into the Siamese Neural Network for training. The model was then tested on another field of view, that was not overlapping, by allowing the same individual or a random person to walk across the frame. This was repeated similarly for all the different types of camera inputs as well. The overall performance of the neural networks were relatively decent, in fact excellent in some cases, based on the extensive testing done and challenges done. That being said, for all the three Siamese Model developed, the area under the curve in the receiver operating characteristic curve were very well in the accepted range, varying from the lowest of 0.64 and highest of 0.80. The average precision recorded in the precision recall curve ranged anywhere from 0.61 to 0.84, which comes to show that all the developed models have done excellently in predicting. Meanwhile, the time taken to train for the 10 epochs were approximately 190 to 310 seconds.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

CHAPTER 1: INTRODUCTION

1.1 Background of Study

Distributed video tracking or Person Re-identification has seen rising interest in this rapidly growing global market, especially in terms of surveillance monitoring. With the increased use of machine learning techniques in wide area of technological fields, it has also seen some heightened potential benefits within the safety and security branches. Monitoring can be defined as a systematic, incessant, active, or passive observation of places, objects, processes, or individuals [3]. Contrarily, surveillance emphasises on individuals, buildings or even vehicles that is targeted or in cases where the subject detected may be deemed as suspicious, based on reliable data.

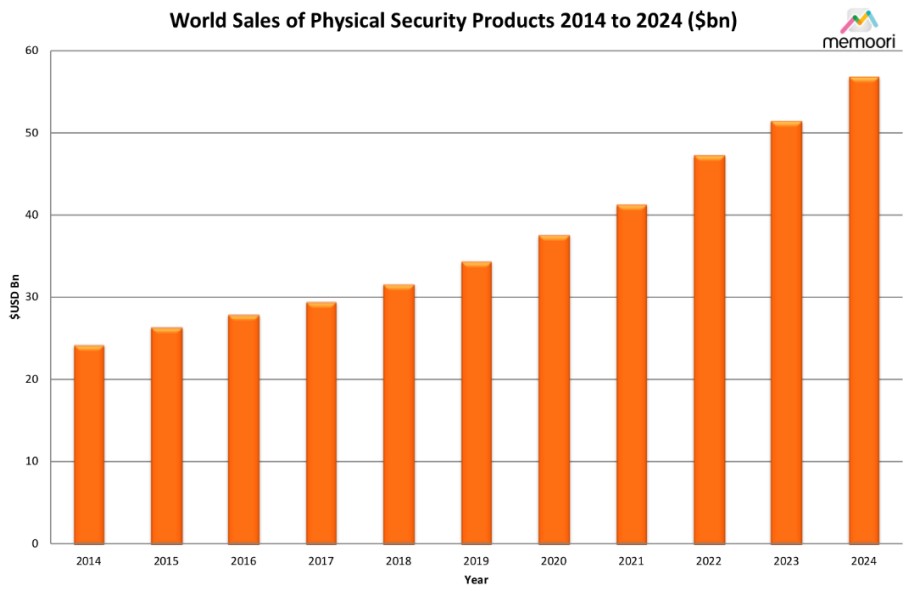


FIGURE 1.1 *– Projectile growth of world sales of Physical security products (2014-2024)*

Physical security has been expanding exceptionally, particularly in the last 10 years as the market has surged to an annual compound growth rate of 6.27 percent. The figure 1.1 above projects that the market would hover around a value of $56.76 billion by 2024 with a compound annual growth rate of an excellent 10.72 percent [4]. With the increasing demand of surveillance monitoring, it is significant to develop more advanced and complex solution which should be more accurate, complex, and cost-effective, thus reducing the requirement of hiring actual supervisors or guards for the jobs that AI could simply handle.

The relatively new implementation of this AI monitoring technique is a widely used technological advancement in liberal countries such as Switzerland, Norway, and Sweden. According to a research conducted [5], artificial intelligence surveillance systems are used by around 51% of advanced democracies. From safe city platforms to facial recognition cameras, democratic governments or countries are employing a myriad of monitoring technology, which does not automatically imply that democracies abuse their systems' rights.

Fundamentally, computer vision tasks can be divided into three branches; Image Classification, Object Localization and Object Detection [6]. For instance, in image classification, the type of classes of objects can be determined and identified with single object inputs such as photographs which, in return, would provide an output of class labels.

The existence of objects can be detected in an image, and the bounding box would be used to determine the position in object localization. The inputs for such cases include photographs with more than a single object whilst bounding boxes with defined heights and widths are the delivered outputs. Similar to object localization, object detection instead would be able to detect each of the bounding boxes' class labels [6].

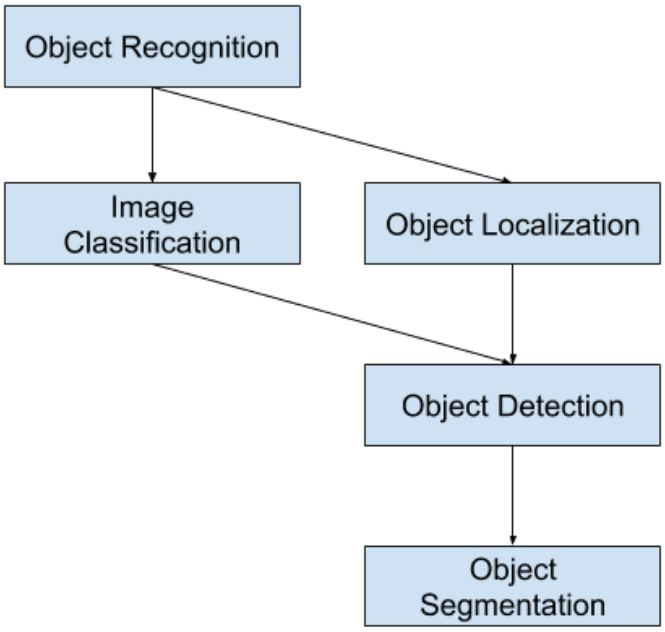


FIGURE 1.2 *– Overview of object recognition tasks*

The figure 1.2 represents the overview of object recognition tasks which can further be broken down to several more subsections. In image classification, algorithms would generate a list of all the object categories identified in the image or frame, while in single object localization provides an additional information of the bounding boxes denoting position and scale of instances in each categories of objects.

The Person Re-identification framework provides a variety of advantages, for instance the most common scenario in which it could be utilised is for video surveillance in security assurance like areas around a shopping mall, parking lot, university campus, or scenarios that involve several non-overlapping cameras. Undeniably, there has been extensive research that has been conducted, that primely focuses in studying the numerous methods of re-identification techniques. Re-ID, this relatively new technology has the potential to observe high growth demand in the upcoming years, due to its extensive beneficial use cases.

However, some of the contradicting issues or problems in distributed video tracking include occlusions, poor lighting conditions, understanding the implementation of colour spaces between different cameras used and appearance variations [7],[8]. Besides, the several methods that exists all provide various sets of percentage performance. Therefore, practicing a suitable re-ID technique is vital in maintaining the accuracy, efficiency, and future deployment of such technology, in the real world.

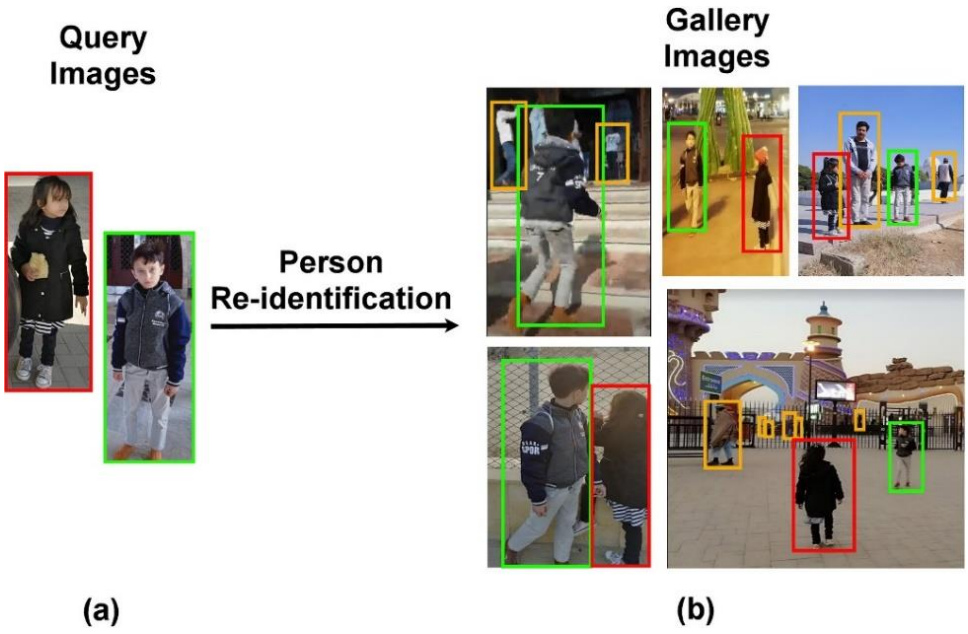


FIGURE 1.3 *– Example of person re-id principle*

1.2 Problem Statement

The Person Re-identification and detection framework in video surveillance allows the tracking of individuals from security cameras; internet protocol cameras, closed circuit television (CCTV). Distributed video tracking, involves challenges that require the retrieval of a specific person from several non-overlapping, discontinuous cameras; attempts to establish whether the person being tracked has appeared in another location or not. This technology, with the aid of artificial intelligence and deep learning mechanism would provide a wide range of advantages that can be utilised in various fields.

Currently, the majority of surveillance monitoring that exists, especially the conventional ones that are out there in the market are limited, due to the dependency of actual human being needed to operate such systems. This caveat that persists is an important factor, because having hired individuals to do such jobs are not necessarily cost-effective, and may lead to unexpected human error. However, with the assistance of distributed video tracking, higher product efficiency may be achieved.

For example, security surveillance monitoring need guards to actually observe whether or not a particular person has appeared multiple times on different, non-overlapping cameras. Hence, guards must closely observe for any suspicious behaviour continuously or view long hours of security footages in case something goes wrong in the future. This would lead to risks or issues such as false identification, delayed recovery, and extended periods of working hours. Some of these are critical scenarios, as it may require a higher budget or time which is not favourable at all times.

Instead, with distributed video tracking and the framework for pedestrian detection would provide smoother yet safer security surveillance options. Since all the information gathered from the cameras are labelled and classified, it would ultimately be saved to a database; can be used to study future research's that may be conducted from the collected data. A well-made and optimised framework, could even eliminate the need of constant physical monitoring, as everything has been done automatically and simultaneously.

1.3 Objectives

The primary objectives for the project are:

- To understand the various methods for capturing parameters for the person of interest.
- To develop algorithms that would be able to detect, track and re-identify the person of interest from non-overlapping cameras.
- To verify and evaluate an existing network in a real-world scenario and on a new data.

1.4 Scope of Study

The project would have an emphasis in understanding the different methods or procedures for capturing parameters on the person of interest. Some of the existing framework for person re-identification that exists are *Comparative Attention Networks (CAN), Part-Aligned Representation (PAR), Deep Contents Aware Features (DCAF), PersonNet, Learning Discriminative Neural Space (LDNS) and Gated Siamese*. The differences and benefits of picking these different methods would be explained further in this report. Besides, this project would cover in understanding the types of different trackers; CNN-based and Correlation filters. This project, however, utilizes the *Faster R-CNN* architecture for person detection, which is simply based on the initial *R-CNN* with major improvements on efficiency and accuracy.

For image processing and the capturing of reference images, OpenCV would be used. It is a programming functions package that focuses mostly on real-time computer vision tasks. TensorFlow would be used for the building the proposed model for the project, as it can be used in a variety of applications, although its focus is primarily on deep neural network training and inference. Both of the machine learning frameworks would be coded completely in Python *(Visual Studio Code as the opted IDE).* Keras, a python-based deep learning API that runs on top of TensorFlow, would allow for the development of more complex deep learning models, and in the case of this project, the Siamese Neural Network.

CHAPTER 2: LITERATURE REVIEW

2.2 Object Detection

With the aid of Convolutional Neural Networks architectures, detecting various objects and segregation according to their classes becomes possible. For example, in a surveillance monitoring system, the developed algorithm should be able to detect individuals and assign respective IDs to them, or a set a count to visualise the number of people are there at any particular instance. Fast R-CNN, Faster R-CNN [19], and Mask R-CNN are more recent architectures that are based on their predecessor, CNN.

2.2.2. Fast R-CNN

Instead of a pipeline, Fast R-CNN is presented as a model that learns and outputs regions and classes instantaneously [21]. Typically, the model would take the image as inputs and generates a collection of area suggestions; processed by a deep CNN network. A distinct layer, Region of Interest Pooling Layer, is added at the conclusion of the deep CNN to extract characteristics relevant to their respective inputs [21]. Contrary to its predecessor, Fast R-CNN inputs the entire image into the Deep ConvNet to obtain the output, Feature MAP [22].

Similarly, *Selective Search* will obtain the bounding boxes for the input images. However, the bounding box will be projected proportionally to the Feature MAP [22], also known as *region of interest (Roi) projection*. Since the fully connected network layer does not work in different sizes, the Roi pooling layer would resize the Roi projection, consecutively classify and predict the bounding boxes.

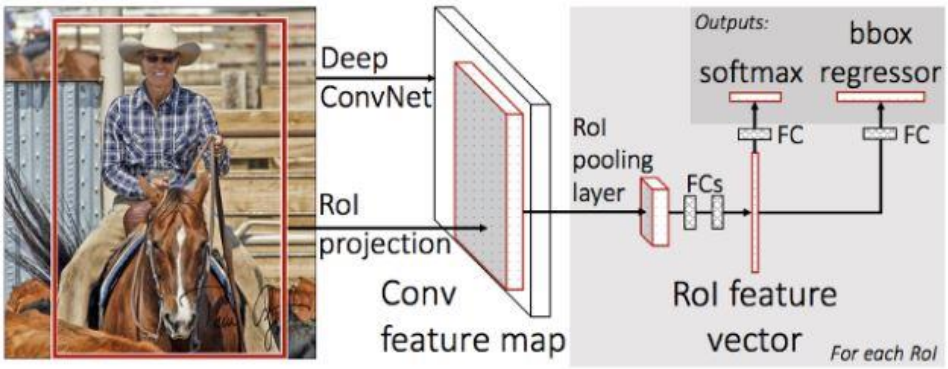


FIGURE 2.11 *– Working principle of Fast R-CNN*

2.2.3. Faster R-CNN

The Faster R-CNN this technique uses the Region Proposal Network (RPN), which is way more cost-effective and computationally cheaper than both its predecessors, R-CNN and Fast R-CNN [21]. The input image would be parsed into the convolutional layer at once and sent to the Feature MAP, similar to Fast R-CNN. Instead of *Selective Search*, RPN would be used for each spatial location [22], [23] for object classification and regressor for bounding boxes. Rather than discovering the region proposals and then passing thru the CNN network for features extraction [13], it merely requires a single time to extract features for the entire region.

Hence, the Feature MAP would then be used to predict the region from the input image. For the purpose of this project, this algorithm would be utilized due to its extremely high efficiency and response time. Despite producing way more accurate results compared to its competitors, Faster R-CNN tend to be computationally intensive and requires high CPU and Graphics processing power usage [29],[30].

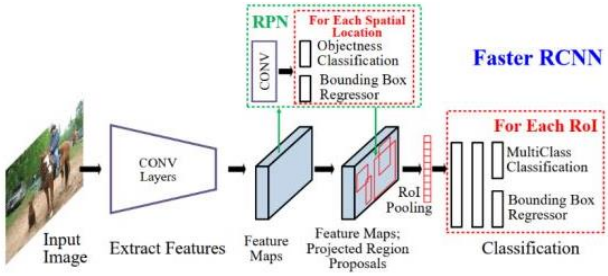


FIGURE 2.12 *– Working principle of Faster R-CNN*

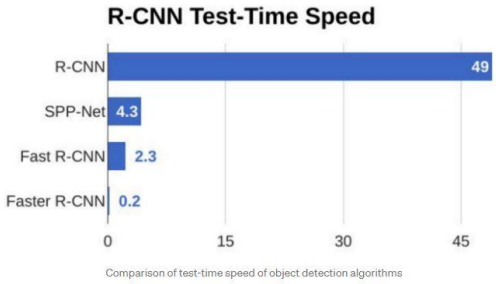


FIGURE 2.13 *–Time Speed tests*

2.2.4. SSD Mobilenet

The Caffe framework was used to implement the SSD MobileNet, which was pre-trained with the PASCAL VOC 2007 dataset [25]. The SSD stands for Single Shot Detector architecture. Since it is pretty accurate, quick, compact, lightweight and has the simplicity to adjust or tune, this framework was excellent for real-time use [26]. However, SSD MobileNet is accompanied with some very obvious limitations; has the ability to only identify and detect 20 different classes. Besides, object categories with relatively smaller size would be harder to detect, because the performance of SSD MobileNet is solely dependent on object dimensions [25],[26].

The SSD framework utilizes a system known as the feed-forward convolutional network to generate a fixed size set of bounding boxes keep tracks of the existence of object classes [30]. Non maximum suppression would then occur which removes the overlapping bounding boxes and reduces the overall noise, before producing the final results [30].

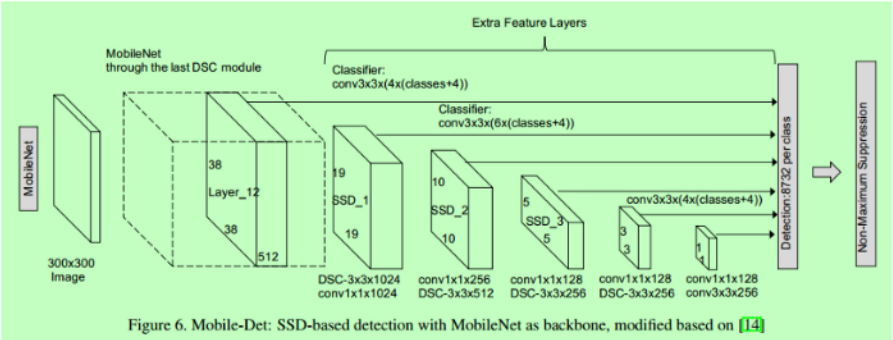


FIGURE 2.14 *– Working principle of SSD MobileNet Framework*

2.2.5. Mask R-CNN

The Mask R-CNN is an extension of the Faster R-CNN framework, but differs by having the addition of a branch that would be used in the prediction of masking a particular object [28]. Now since the architecture is able to generate masks in pixel-level, it would be an easy task to perform other more complicated jobs, like predicting various human postures [29]. On a myriad of tasks, Mask R-CNN can outperform all single-model competitors that exists as of today [29]. However, being able to execute such intricate challenges comes with a price; tend to be computationally intensive and requires longer processing time [29].

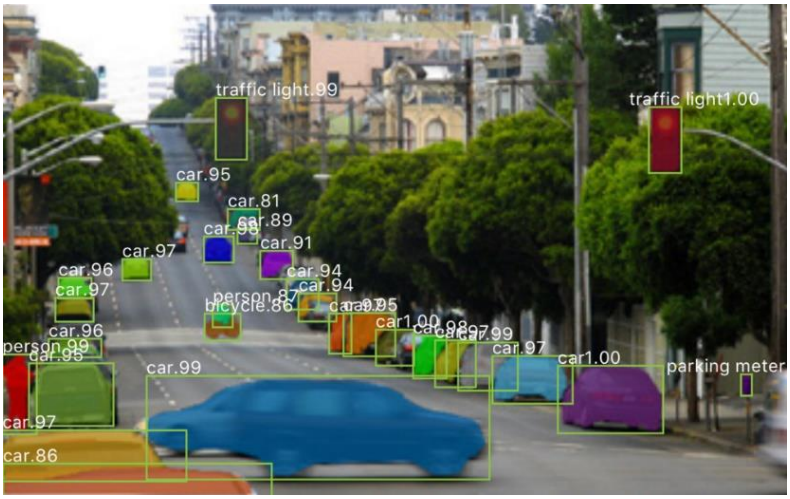


FIGURE 2.15 *– An example of the Mask-RCNN Framework*

2.3 Object tracking

This deep learning application utilizes a set of tools to collect the initial objects' observation which allows the identification and creation of different classes of objects as they move across frames of a particular video. It would then assign respective IDs to each of the identified items. In a more define manner, object tracking accurately detects objects in a video and understand them as a collection of trajectories [16]. Object tracking in machine learning can be divided into two different categories [13]; CNN-based and Correlation filters. Despite being only slightly better than the Correlation filters, CNN-based approaches tend to take longer time and requires prior training. Contrarily, DCF-CSR (Discriminative Correlation Filter with Channel and Spatial Reliability) [18], provides much accurate results whilst being a durable tracker that is efficient enough to run in real-time.

2.3.1. Centroid Tracker

This tracking algorithm is highly effective; the way it works is, it depends on the Euclidean distance among existing object centroids and new object nodes [31]. It does this by identifying the centroid of the bounding boxes to which new IDs are assigned [31],[32]. Prior to these steps, object detection should have been performed on each of the corresponding frames. Between each of these frames, the centroids must be near together for it to work efficiently. Due to its extreme simplicity, centroid tracker would not be able to perform well while tracking a number of objects in real-time.

Centroid tracking merely consists of a few steps; head-starting with a series of object detections, and for each of the respective first detections, unique IDs would be created. Hereafter, the respective IDs should be maintained as the objects move across frames, without assigning a new ID for the particular object, based on the limitations set.

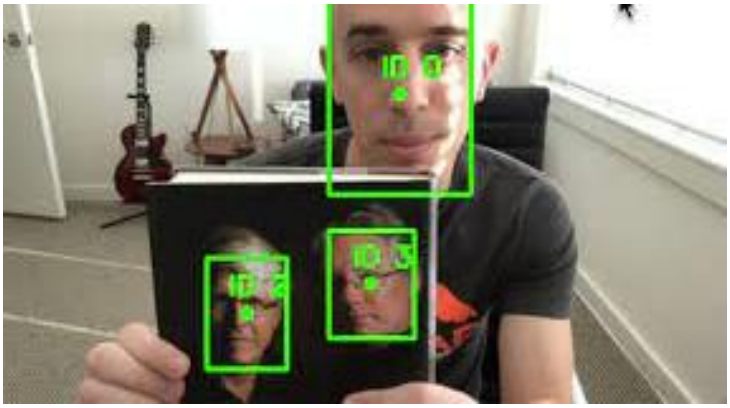


FIGURE 2.16 *– Assignment of IDs based on detected individuals*

Centroid-based object tracking can be divided into four main steps that allow the assignment of IDs, as it is requires multi-step procedures;

- Determine centroids by identifying bounding box coordinates.
- Calculates the distance between the newly formed bounding boxes and the already ones in Euclidean space.
- The coordinates of the existing objects would constantly be updated.
- The registration of the new objects (assigning IDs)

2.3.2. SORT (Kalman filters)

SORT (Simple Online and Realtime Tracker) is a framework that is based off Karman filters [33],[43]. It is an online tracking algorithm which make predictions regarding the current frame, and it takes into account information from the current and prior frames [34],[35]. Associating items in two adjacent frames provides the rectification information; by using attributes and functions like IOU (intersection over union). It's intended for Realtime tracking using only previous and current frames, and instantaneously generate object identities. Each bounding box that represents an item is allocated a unique track ID; confidence value greater than a certain threshold [35].

SORT would not be able issues like occlusion and re-entering of objects [42], resulting in the development of a more complicated model that utilizes deep learning, known as DeepSort [44],[45]. However, unlike SORT, DeepSort requires high CPU and GPU usage. Hence, the hardware necessary to perform a more complicated technique at a higher frame rate is prohibitively expensive [35]. For the purpose of this project, SORT has the potential to get object tracking sorted out due to its light-weighted nature, hence easier integration with a more hardware intensive object detection module like Faster R-CNN [36].

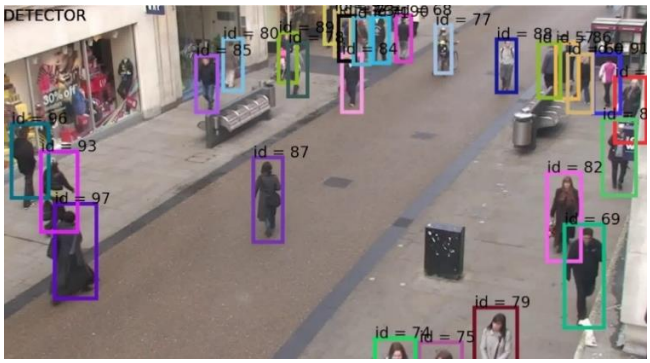


FIGURE 2.17 *– Example of the implementation of SORT*

2.3.3. DCFnet

DCFnet is a network architecture that is light in weight, allowing the tracker to run at a high speed of above 60 frames per seconds theoretically. Besides being highly compact, dense and quick. The term DCF is an acronym for Discriminant Correlation Filters, introduced as a particular correlation filter in a Siamese Network [36]. The problem with this deep learning online algorithm is, that much more conceptually sophisticated and complicated than its competitors of the same range [36].

The Discriminant Correlation Filter (DCF) is then derived backpropagation via it by characterising the network output as a likelihood heatmap of a particular object position. The effectiveness attribute of DCF is intact since the derivation is still done in the Fourier frequency domain.

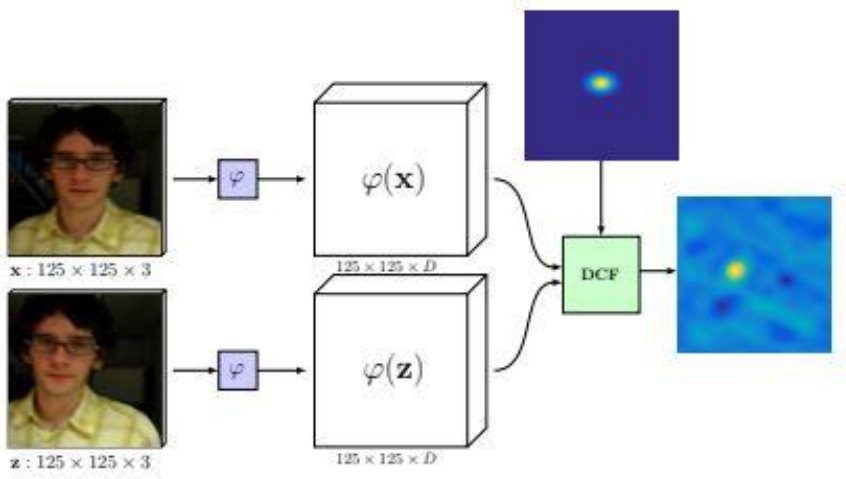


FIGURE 2.18 *– DCFnet architecture*

2.4 Siamese Neural Network (One-shot Image Recognition)

The Siamese Neural Network, allows the build of models that are not particularly achievable with the use of pure classification. This deep learning neural network is made up of two subnetworks that each produce two embeddings. Those extracted features are then sent into a loss function as inputs. In terms of functionality, the Siamese Model is barely distinct than most convolutional neural networks that exists; inputs images, and converts their characteristics into a numerical representation. The distinction is in the post-processing of the result [41].

Hence, from the characteristics and features collected in the first step, the metric learning stage would then determine the similarities of the two images. If the distance is smaller between these two compared images [13], it can be predicted that the chance of detecting the same person is higher. In [14], a conventional person re-identification framework however, would first include pedestrian detection, followed by extracting the features of individuals, some level of photometric and geometric transforms, and to increase the accuracy, similarity estimation would be concluded, as seen in the figure below.

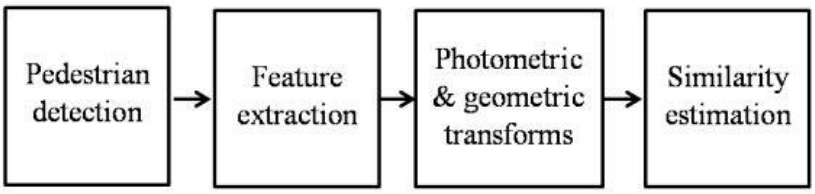


FIGURE 2.19 *– Typical flow of person re-identification [14]*

For further optimising coverage areas, surveillance systems in public places typically employ networks of cameras. However, due to financial and infrastructure constraints, these cameras frequently have limited or no functionality; no overlap within the area of view, hence identifying individuals across various other cameras is a hefty process [9]. Some of the pre-existing frameworks developed for Person Re-identification can generally be divided into two contrary subcategories; traditional and deep learning methods. Hence, this is where a model like the Siamese neural network comes in handy.

The Siamese model can be summarised as follows [40]; create a model that is able to distinguish between a set of similar and dissimilar pairings of images, and for verification, generalise to identify new classifications based on learned feature mappings. This deep learning approach, captures attributes to transformations in the input vector by employing a framework with a set of parameters and layers of non-linearities. The figure below represent the following statements, where the model should be able to identify the closest image related to, from an input image.

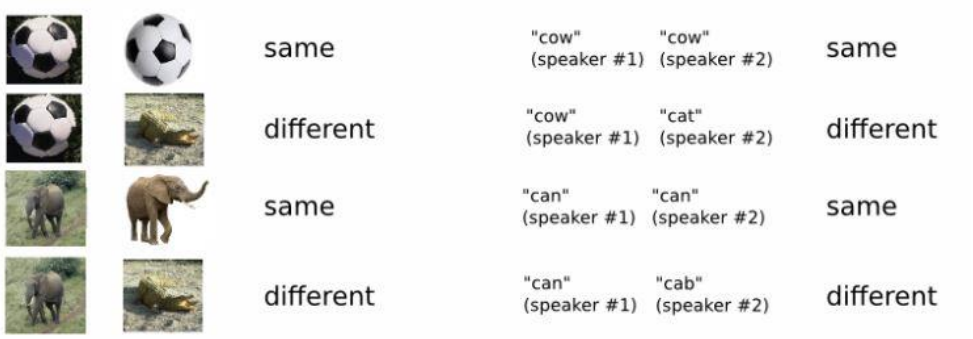


FIGURE 2.20 *– Verification of tasks during training [40]*

The framework is made up of a series of fully connected layers, each of which employs a single channel with different-sized filters and a constant stride of one [40]. Rectified linear activation function (ReLU) is embedded in the neural network. It is a nonlinear linear function that would output the input if the detection were positive, else it outputs zero [42]. The framework proposed in [40] suggests the input image to undergo convolution, ReLU and max-pooling for three consecutive times, before fully connecting the feature maps with sigmoid and L1 Siamese distance.

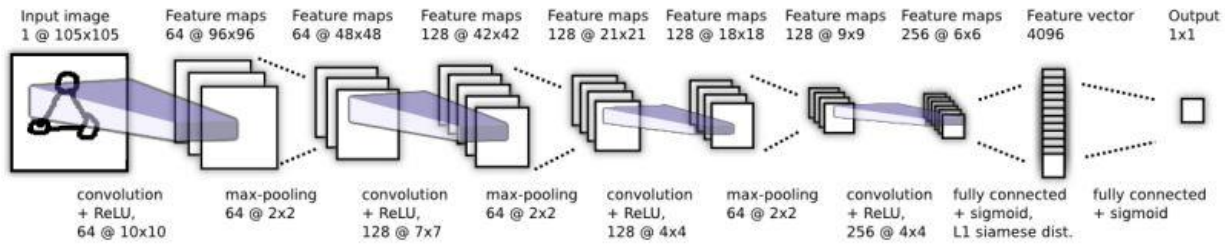


FIGURE 2.21 *– One-shot Image Recognition [40]*

2.5 Non-Maximum Suppression (NMS)

NMS (Non-Maximum Suppression) is a key tool in a myriad of object detection algorithms, which drastically increases the accuracy of object localization. In object detection output, there are a significant number of bounding box candidates, and these redundant bounding boxes impair detection accuracy [37]. NMS is used by most object detection algorithms to reduce a large number of detected bounding boxes to only a few. Windowing is used by most object detectors at the most basic level. Hundreds of thousands of windows of varying sizes and shapes are created. These windows are said to contain only one object, and each class is assigned a probability/score by a classifier. After the detector generates a huge number of bounding boxes, the best ones must be filtered away. Hence, the most often used algorithm for this task is NMS.

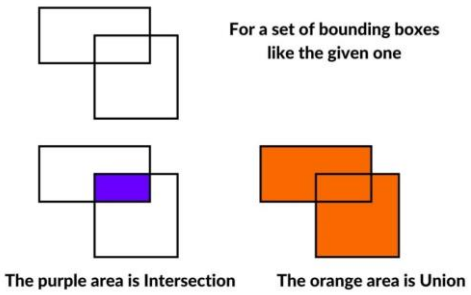


FIGURE 2.22 *– Working principle of the NMS*

NMS is often employed at the end of the process to reduce the number of detections and establish the bounding box with the highest confidence level [38]. The Jaccard index, also known as the Intersection over Union (IoU) metric, is a method for quantifying the percentage overlap between both the ground truth Bounding Box and the predicted Bounding boxes [39]. With the following the steps, there should be only a single bounding box that appears, which is also the most accurate one.

FIGURE 2.23 *– Removal of noise (additional bounding boxes)*

CHAPTER 3: METHODOLOGY

3.1 Project Proposed Method

The project, Distributed Video Tracking will take real-time video feed from the internet protocol cameras. It is then divided into two main algorithms; the first algorithm is catered for Pedestrian detection and tracking whilst the second algorithm focuses in reidentifying the pedestrian.

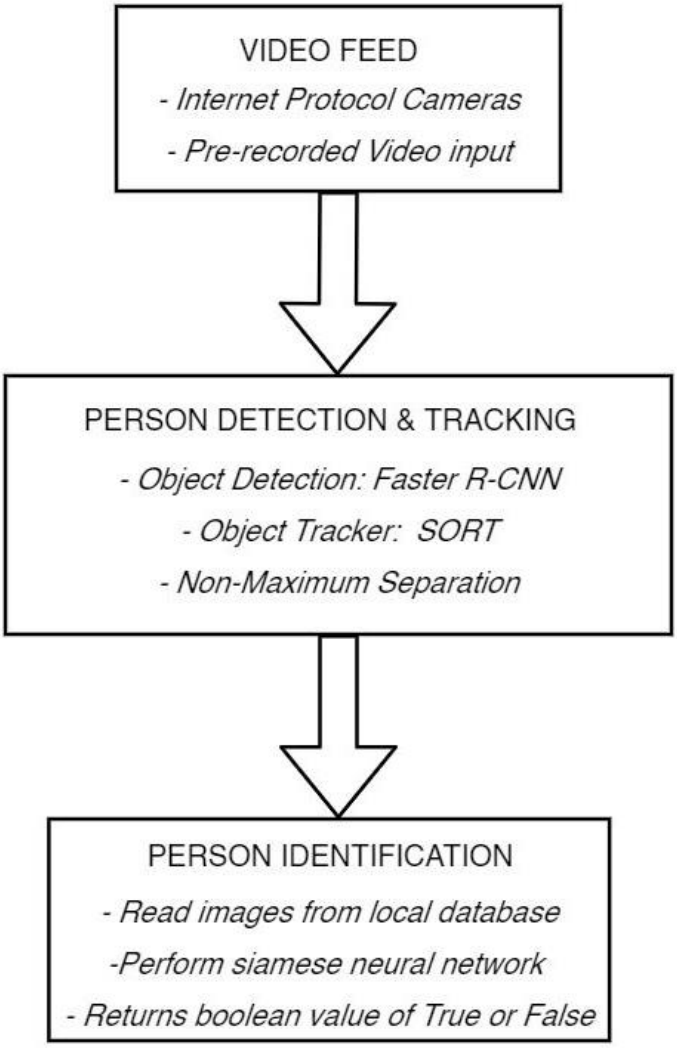


FIGURE 3.1 *– Project Block Diagram*

3.2 Project Methodology

i. Algorithm 1: Pedestrian Detection and Tracking

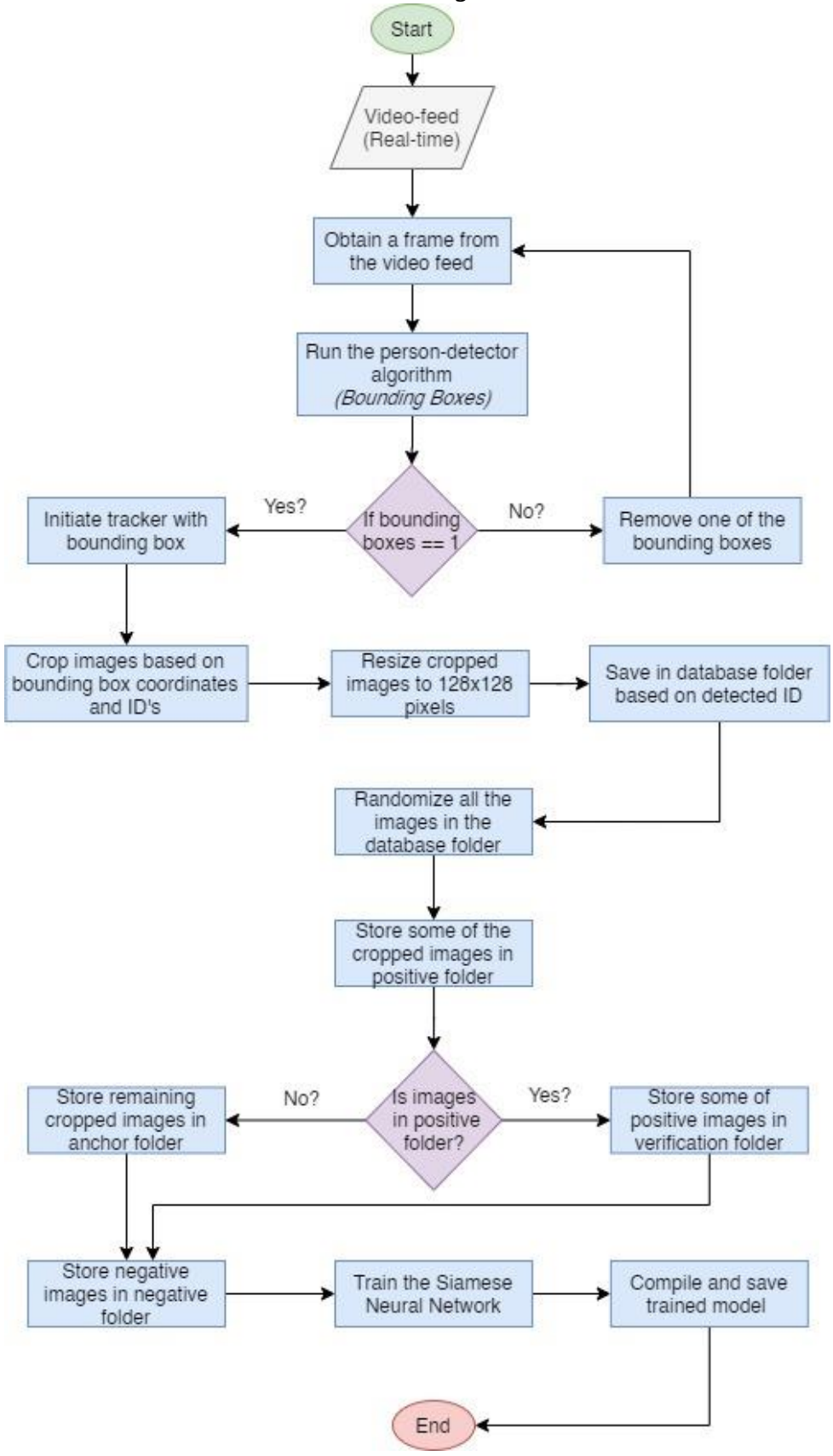


FIGURE 3.2 *– Pedestrian Detection and Tracking*

ii. Algorithm 2: Pedestrian Identification

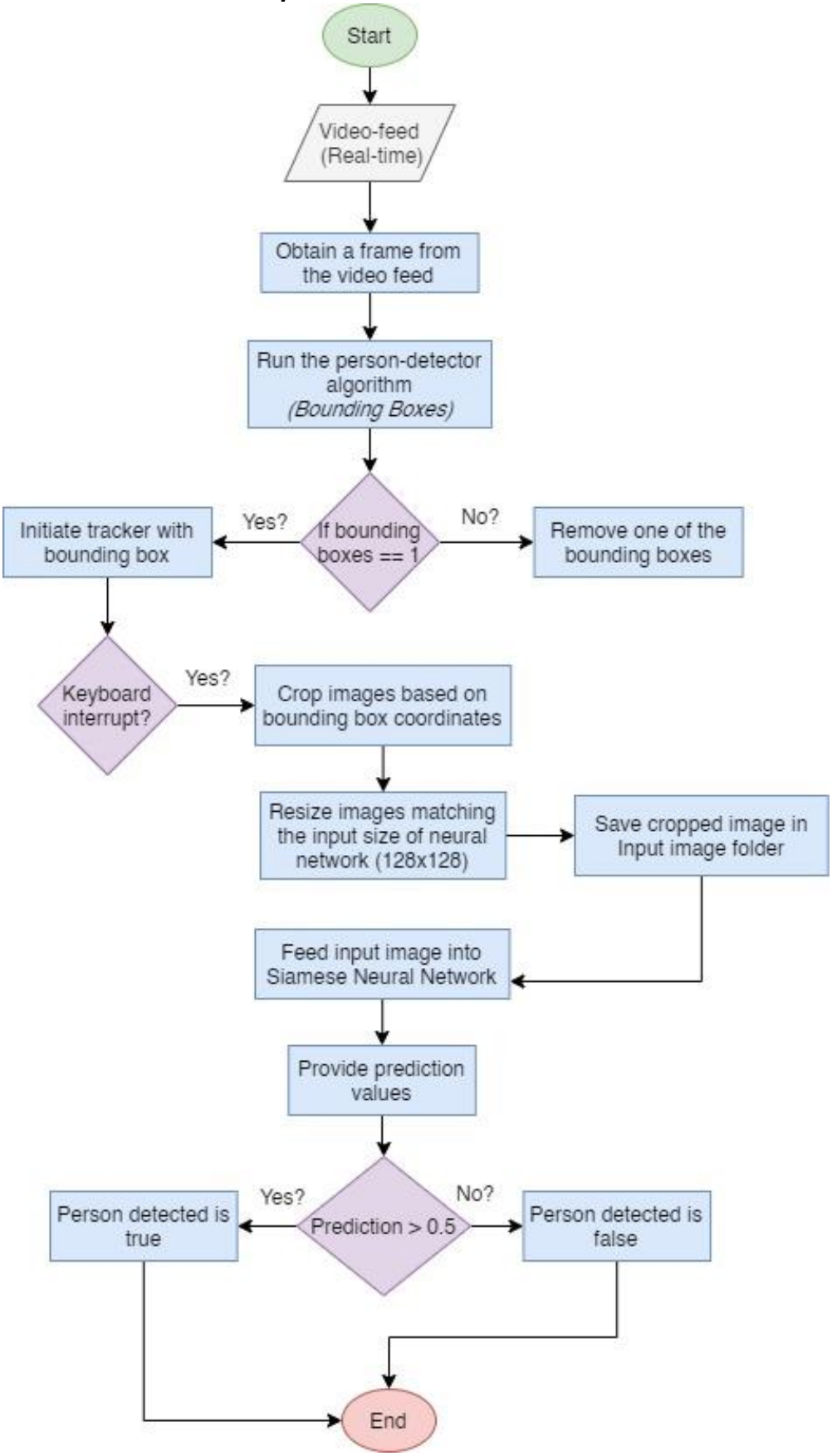


FIGURE 3.3 *– Pedestrian Identification*

3.2.1 Pedestrian Detection and Tracking

The *Pedestrian Detection and Tracking* algorithm compromises of three main subsections; object-detection, non-maximum suppression, and object tracking. The system would obtain a frame from the video feed (real-time footage from the Internet Protocol cameras) to run the person-detector algorithm. For this, the Faster R-CNN object detection API will be utilized; only selecting the *"person"* class. This would allow the object detector to only identify individual(s) in a continuous frame, ignoring other object classes during detection. Once the person is detected within the video feed, the non-maximum suppression step would be performed. From the obtained bounding boxes, if the number of bounding box is more than one, the remaining or the additional (not required) bounding boxes would be removed, and the step of obtaining a frame from the video feed is repeated. This is a significant step as it removes overlapping bounding boxes or noises when the *"person"* class is detected.

Next, the tracker, Simple Online and Realtime Tracking (SORT) would be initiated and through SORT, the identified bounding box coordinates and their respective IDs would allow for to crop the images of individuals across the frame. It would be resized into 128x128 pixels, and saved in a database folder based on the detected ID. The images would then be randomized and some of the images would be stored into the positive folder. The remaining images (must not be the same as the ones in positive folder), would then be stored into the anchor and verification folder respectively. Then, the negative images, which are images retrieved from a random dataset (CUHK01 or PKUv1a_128x48) would be stored in the negative folder. Hereafter, the data is prepared, and can be used for training in the Siamese Neural Network. The trained model would then be compiled and saved for future purposes.

In Siamese Neural networks, the input image of a detected person is considered to be the anchor image. Assuming there is the anchor and positive image pair, and the anchor and negative pairs, convolutional neural network would then obtain the similarities between these two images. The positive images, are supposed to be the images of the same person (preferably with a different feature), whilst the negative images are images of a different person. Through this, the Siamese network can easily identify whether or not there are the same person, by encoding the feature representation of the given two images (negative and positive). This is also known as the triple loss function.

3.2.2 Pedestrian Identification

The *Pedestrian identification* algorithm has relatively short procedures, which are more sophisticated to perform. Similarly, the system would obtain a frame from the video feed (real-time footage from the Internet Protocol cameras) to run the person-detector algorithm. Once the person is detected within the video feed, the non-maximum suppression step would be performed. From the obtained bounding boxes, if the number of bounding box is more than one, the remaining or the additional (not required) bounding boxes would be removed, and the step of obtaining a frame from the video feed is repeated.

The tracker, SORT would then be initiated, and the bounding box coordinates will be identified. The system would then continue to display the video input, whilst waiting for a keyboard interrupt. If there was a keyboard interrupt, the system would crop images based on the bounding box coordinates, resize them to the input size of the neural network of 128x128 pixels. The images are then stored into the input image folder, and fed into the Siamese Neural Network. For this project, the implementation was at 128x128 pixels; square dimensions due to the way the model was built upon. As mentioned previously, with an input image of 128x128 pixels, the Siamese model would resize it to 105x105 pixels, before performing the typical steps involved in a neural network; *convolutional*, *maxpooling*, *flatten* and *dense*.

The input image would then be compared with the verification images, and hence provide the prediction values. Once the input image instance is fed into the network, the prediction probabilities are made. For probabilities that have any values that are above 0.5, predictions are classified to 1 and 0 for probabilities lesser than 0.5. This ensures that the person of interest detected in a different camera or field of view is the same exact person; returns a Boolean value of **True** or **False**.

3.3 Software, Hardware and Tools

Software, Hardware and Tools	Name	Description
	TP-LINK R1043ND	• In charge of internet traffic direction • Allows the configuration of internet protocol cameras (static IP addresses)
	D-LINK DCS5030L	• Obtains the video footages in real-time • Supports HTTP stream instead of RTSP • Connected to a router • Configured to static IP addresses
	VISUAL STUDIO CODE (PYTHON)	• An IDE that utilizes Python, which allows to program the whole system. • The ability to transition between Python environments with ease

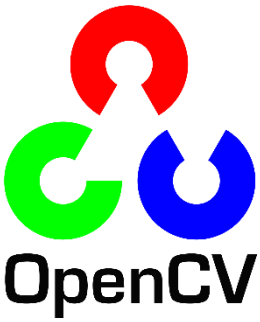
	OPENCV	• Would be used to configure various models from TensorFlow • Take advantage of hardware acceleration and multi-core systems to deploy • Strength are in the deployment side
	TENSORFLOW	• TensorFlow to build, test and train the proposed model • Scalable; allows its users to develop any kind of system using TensorFlow • Strength are in the training side
	KERAS	• Various backend and modularity • Multiple GPU support • Has pretrained models

TABLE 1 *- Software, Hardware and Tools*

3.4 Gantt Chart

FYP I												
Activities/Week	**W1**	**W2**	**W3**	**W4**	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W11**	**W12**
Obtain & understand the topic advised												
Perform background study, objectives, and problem statement												
Literature review on the different ML/DNN models												
Receive required hardware; IP Cam/Router/Switch												
Selecting the tools and ML/DNN Models												
Deploy Object Detection API and Tracking.(Algorithm I)												
Testing on the different object detection models (Faster R-CNN and SSD MobileNet)												
Testing on the different trackers (Centroid tracker/DCFnet/DeepSORT)												
Developed simple model to crop images from video input based on the coordinates of detected bounding boxes												
Result Analysis and further testing												
Documentation												

FYP II												
Activities/Week	**W1**	**W2**	**W3**	**W4**	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W11**	**W12**
Identify approaches to develop re-identification network												
Switching from Centroid tracker to SORT for object tracking												
Testing on transfer-learning approach on Faster R-CNN model for custom object detection												
Develop the Siamese Neural Network (Algorithm II)												
Integrate the two developed models (Algorithm I and Algorithm II)												
Result Analysis and further testing												
Improvement and optimisation to reduce false detections												
Documentation												

CHAPTER 4: RESULT & DISCUSSION

4.1 Initial Phase

The Faster-RCNN Inception v2 object detection API from TensorFlow was used for in detecting individuals or person in a particular frame. The Inception V2 architecture is used since it is one of the most accurate Convolutional Neural Network (CNN) architectures that exists. Kernels of various sizes are required for successful identification of such a variable-sized feature. That is exactly what Inception accomplishes. It expands rather than just becoming deeper in terms of the number of levels. Within the same layer, many kernels of various sizes are implemented. Despite requiring high graphical power usage, it is able to provide extremely accurate results in identifying and classifying objects detected from a video input.

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

FIGURE 4.11 *– Comparison between the different Network architectures*

In this project, object detection APIs were imported from TensorFlow for them to be deployed on OpenCV. However, in order to integrate object detection models from TensorFlow, OpenCV requires an additional configuration file. Similarly, as in TensorFlow serialised graph formatted as protocol buffers (protobuf) would be used, OpenCV relies on a text version of it. Besides, Cuda and cuDNN was installed for high-performance GPU acceleration, based on their respective versions, to ensure compatibility with TensorFlow. For Tensorflow 2.8.0, cuDNN 8.1 and CUDA 11.2 were required, while ensuring python 3.x was already installed as base environment. This ensures GPU hardware support during rendering.

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.7.0	3.7-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1

FIGURE 4.12 *– TensorFlow versions and their compatibilities*

The Internet Protocol Cameras (D-LINK DCS5030L) obtains the video footages in real-time. It supports HTTP stream instead of RTSP, and needed to be connected to a router for the static IP addresses configuration. Initially, the camera were connected via Wi-Fi of the same network, streamed with HTTP. This resulted with the issue of reassigning new IP addresses, each time either the cameras or router restarts, as several additional steps were needed to obtain the live stream on the system for further processing. To eliminate this, the cameras were configured to a specific IP address and mapped to their corresponding MAC Addresses in the router configuration settings. For B0-C5-54-2E-2E-E7 it was bind to 192.168.0.30, while B0-C5-54-2D-41-94 to 192.168.0.40. These two cameras were connected to TP-LINK R1043ND router; In charge of internet traffic direction and allows the configuration of internet protocol cameras (static IP addresses).

FIGURE 4.13 *– DLINK-DCS5030L configuration page*

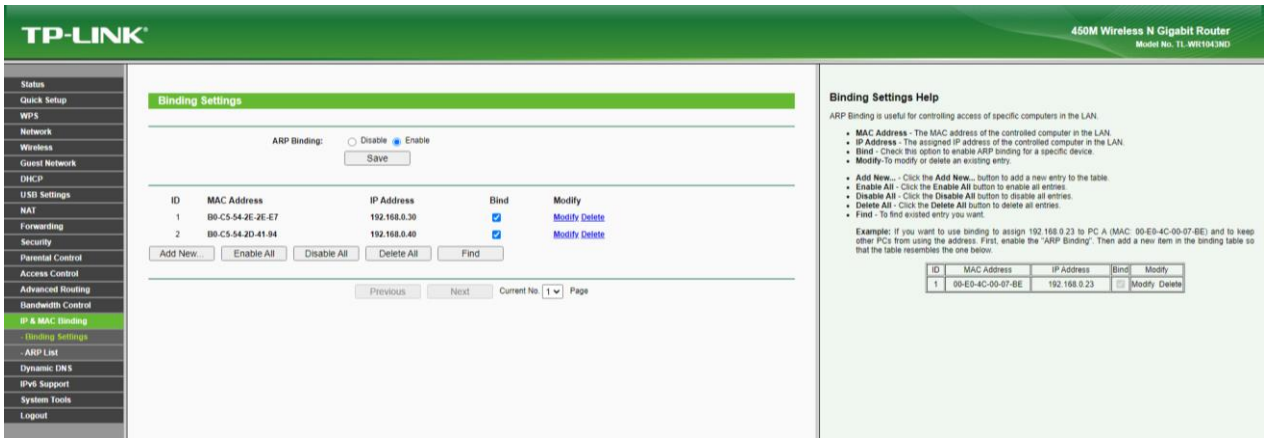


FIGURE 4.14 *– TP-LINK R1043ND router configuration page*

4.2 Person Detection and Tracking

The hardware specifications of the computer used throughout the testing were; NVIDIA GTX 1060 with 6GB (gigabytes) of VRAM. This was made possible due to CUDA, which enables applications to use specific types of graphics processing units. Three inputs were used to test the overall performance of the Faster-RCNN Inception v2 in detecting *"person"* class; two pre-recorded videos with distinct recording time at 60fps, and a live stream through HTTP at 30fps with the D-LINK DCS5030L IP cameras. Two of these video inputs all have an individual appearing in and out of the frames, albeit from different angles, light setting, and locations, which challenges the effectiveness of the object detection model.

The Faster-RCNN Inception V2, accompanied by the Simple Online and Realtime Tracking (SORT) would work simultaneously to detect a person moving across the frame. This would work regardless on the type of input, whether it is from a pre-recorded video, or a HTTP stream. As mentioned, the identified bounding box coordinates and their respective IDs would allow to crop the images of individuals as he moves across the frames in the video. It would then be resized into 128x128 pixels, and saved in a folder based on the detected ID. It is also to be reminded that the names of the cropped images would be titled as follows; ***"path + str(i)+ '_FRAME_' + cur_strg +'.png'"***.

The number of frames and detected ID play a significant role in segregating the cropped images based on the ID identified. If there are more than a person detected in the video input, then the ID would be classified based on the detected ID and placed into their corresponding folders. The figure 4.15 below, demonstrates the working principle of the algorithm described, highlighted in yellow.

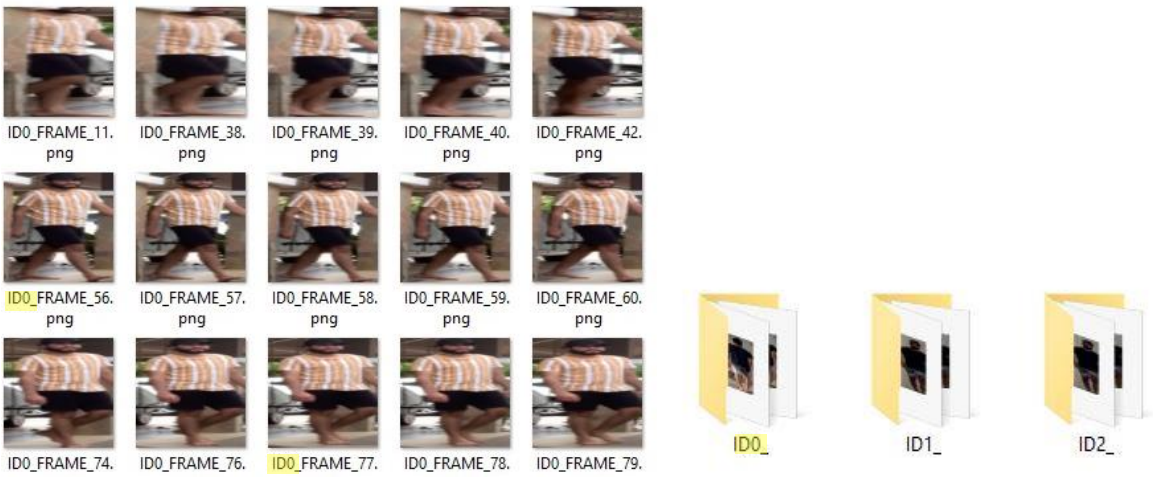


FIGURE 4.15 - Demonstrates the working principle of the algorithm

Figures 4.16 and 4.17, SORT detects a person as ID_0, which would result the images that are cropped to be placed in a folder respective to the detected ID. So, in this particular case, the cropped images are all stored in a folder named “ID_0”. The cropped images are then resized to 128x128 pixels to ensure the standardization of all images (positive, anchor and negative), before being fed to the Siamese model. The neural network was built to only receive square images of 105x105 pixels, which quickly translates to being able to identify images that have sides that are parallel and equal in dimensions.

FIGURE 4.16 *– Video input from a pre-recorded video(1)*

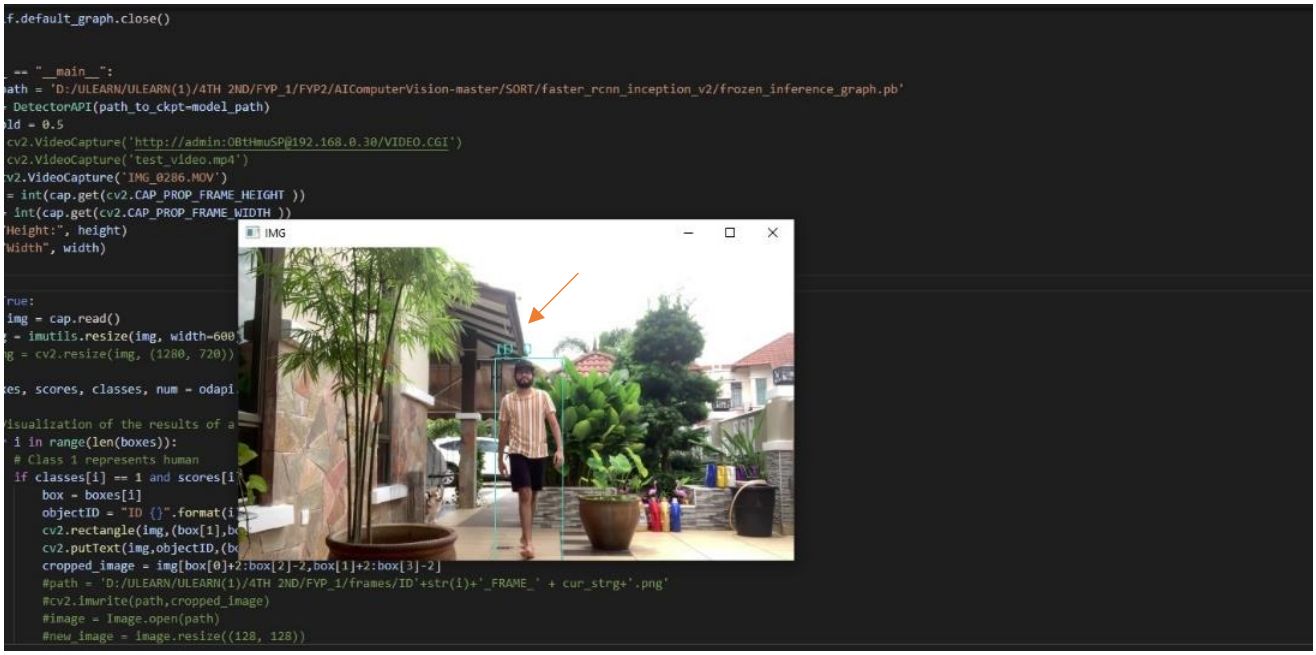


FIGURE 4.17 *– Video input from a pre-recorded video(2)*

Once the training is complete, another recorded video, preferably from a different angle was used to test the legitimacy of the training results. But first, it would be ideal to perform a simple experiment to test the Siamese Model on the video used for the training itself. Figure 4.18 shows the video that was used for training in performing the verification, and that the results detected are mostly positive (seen in the bottom left, highlighted in yellow). This means that the Siamese Model was able to identify the person as the same person, that was used as the anchor and positive images.

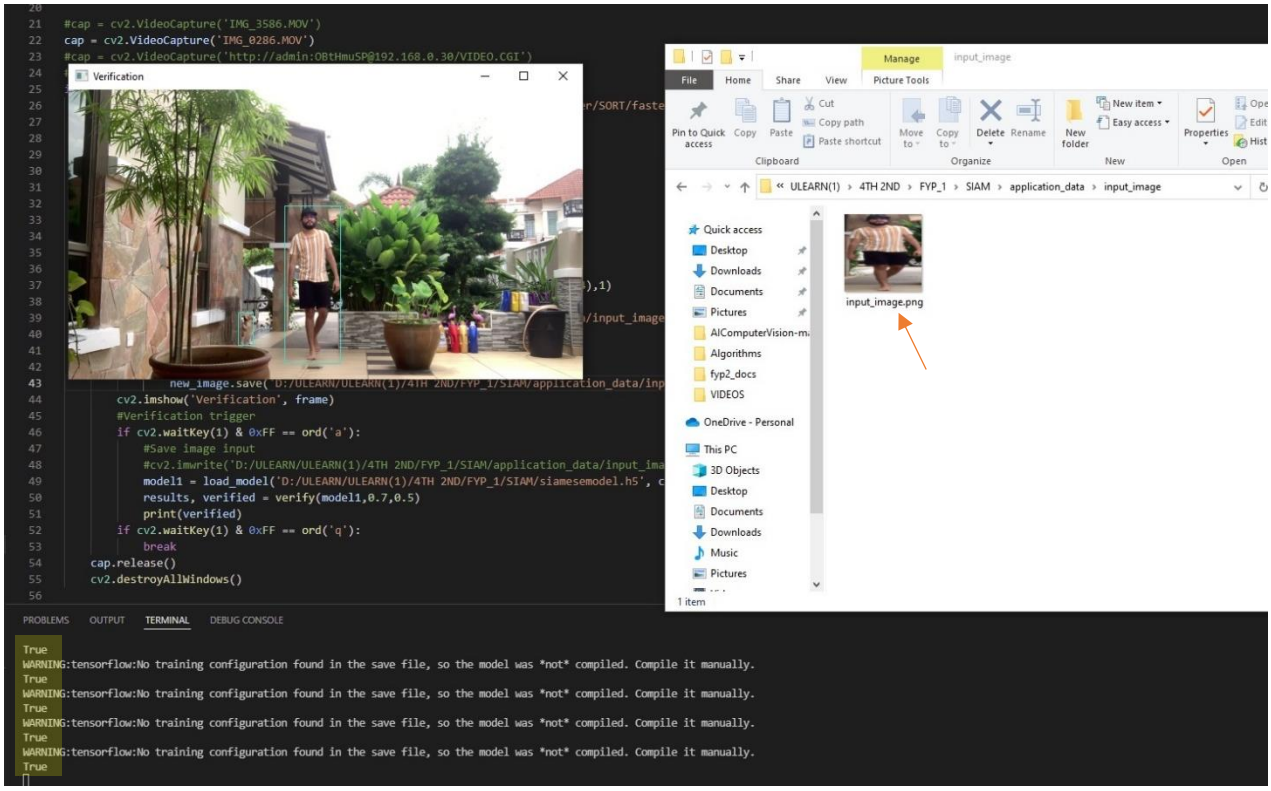


FIGURE 4.18 *–To test the Siamese Model on the video used for the training itself*

On the right of Figure 4.18, shows the input image that would be used for verification. This is also the cropped images that are based off the bounding box coordinates. The input image would then be saved as ***input_image.png***, which can be triggered by a keyboard interrupt. This is simply an instance of the image, meaning, that each time there is a keyboard interrupt, the next detected bounding box would replace the current cropped image. The ***input_image.png*** is basically the image that would be fed into the neural network where the probabilities would be predicted. The test were done with a verification threshold of 0.7; any predicted probabilities of over 70% would return a Boolean value of **True**, and anything below as **False**. As can be seen, the model does a pretty decent job in identifying the same person.

The test was then mimicked on two random videos, simply to confirm that the Siamese Neural Network developed did not fake the results obtained. The figures 4.19 and 4.20 below shows that the person detected is not the same person that was used in training, or in other words the anchor and positive images. Consecutively, the result would be **False**, classifying that this particular person is not recognised.

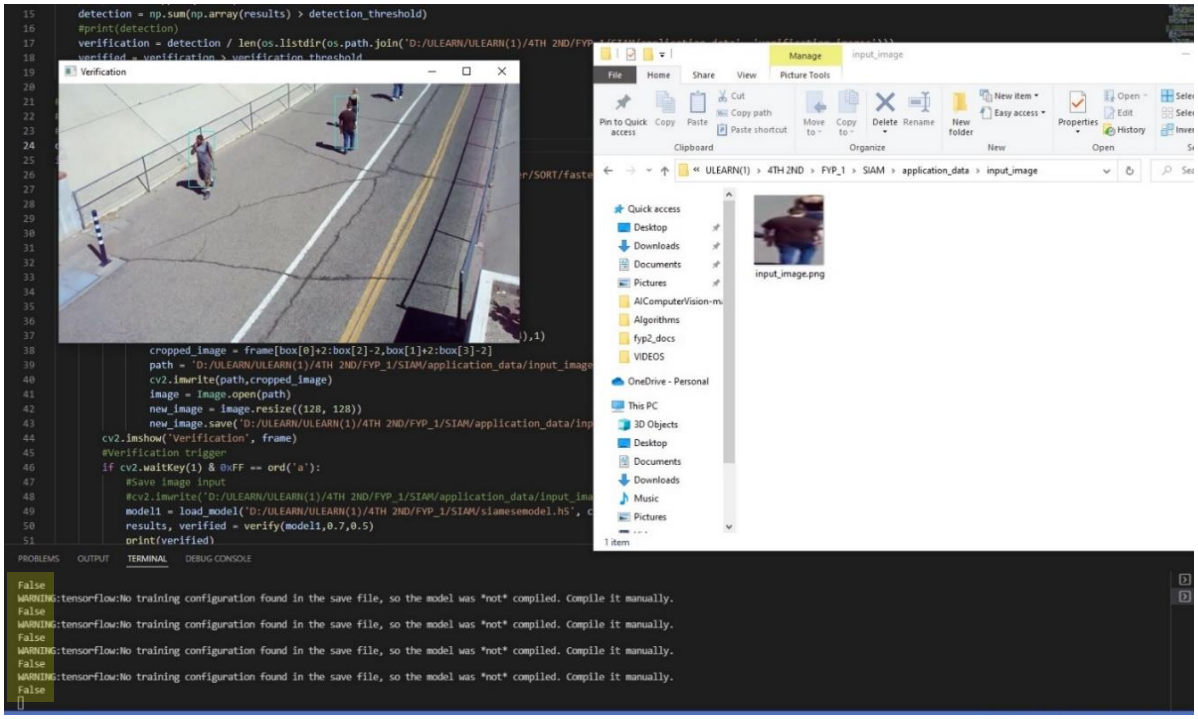


FIGURE 4.19 *– Random video input (1)*

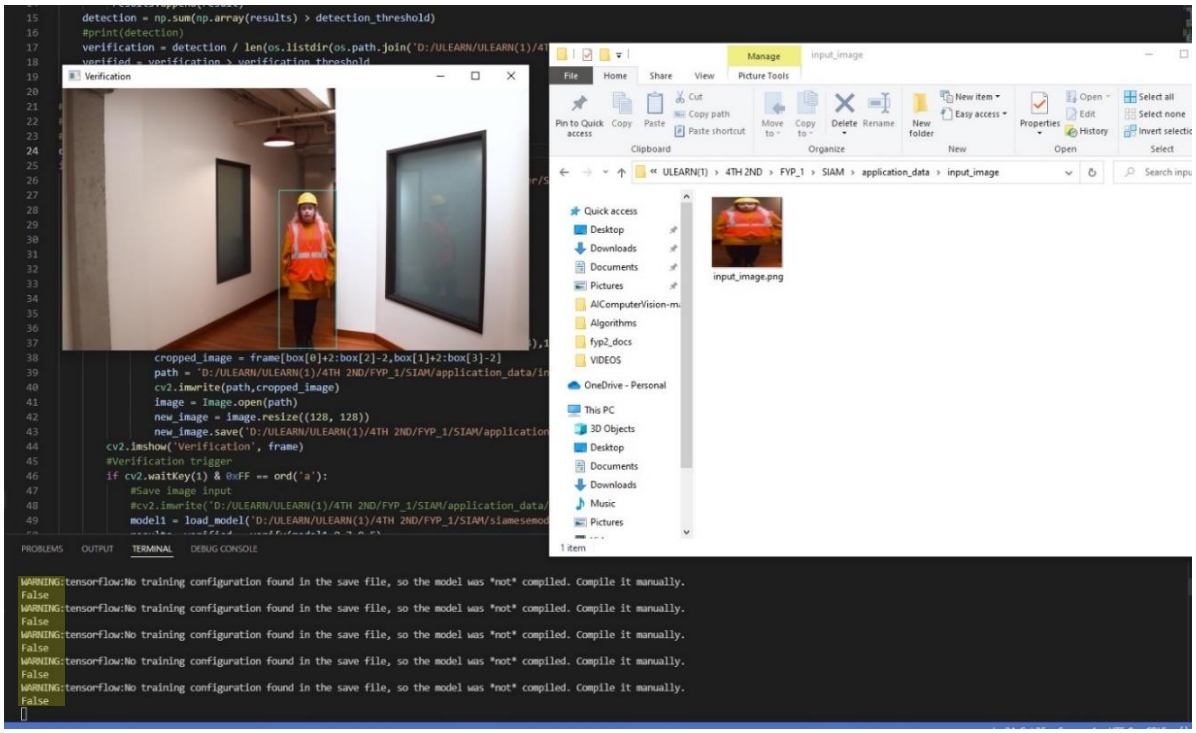


FIGURE 4.20 *– Random video input (2)*

This time, however, the second video input simulates a different camera angle, of which the same person walks across the frame. Take a look in the figures 4.21 and 4.22 below. Unsurprisingly, the model was able to detect the individual as the same person detected from the first video input, and hence returns a Boolean value of **True**. The Siamese Model satisfactorily produces results that was able to classify that this particular person, was indeed the same exact person from the first video input, albeit from a different camera perspective.

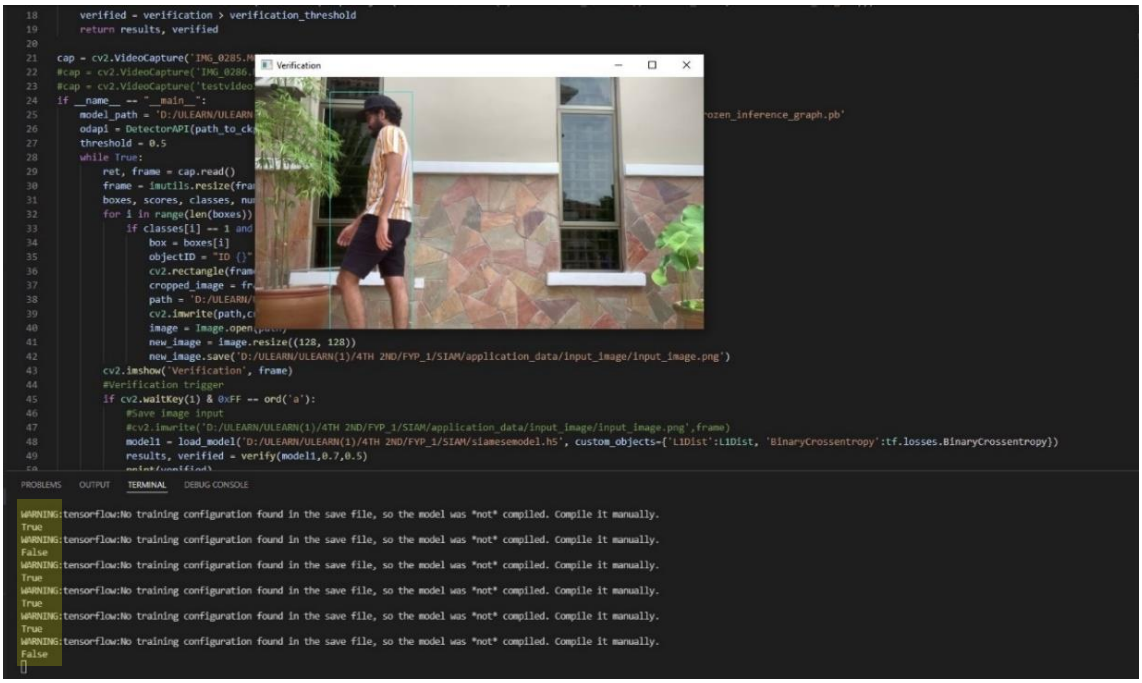


FIGURE 4.21 *– A pre-recorded video, from a different angle (1)*

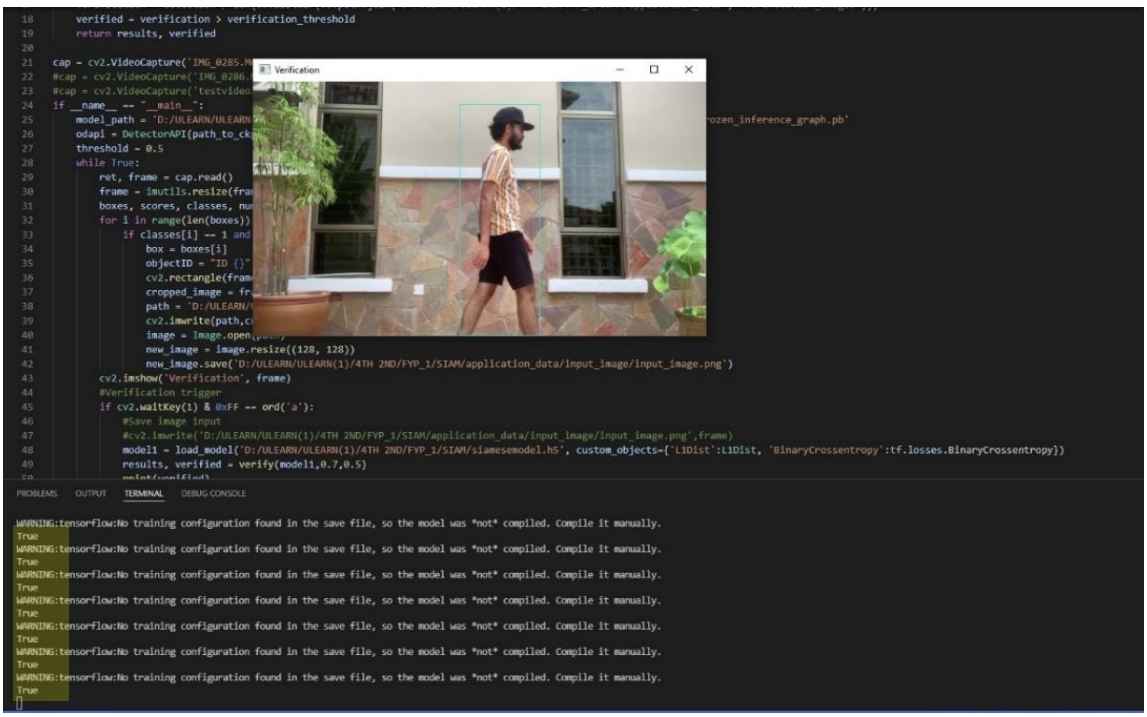


FIGURE 4.22 *–A pre-recorded video, from a different angle (2)*

The frame rate is the number of pictures collected or shown in a row. The average frames per second recorded from these two-video inputs were approximately, 2.521 and 2.477 respectively. Undeniably, these results ridiculously slow, despite the fact that there were from inputs of already recorded video. The primary reason to this is due to the fact that both the object detection model, Faster-RCNN Inception V2 and the tracker, SORT are extremely GPU intensive. The system was run on a satisfactory and relatively mid-range NVIDIA GPU, causing the overall running speed to be time-consuming.

The figure 4.23 below represents the average fps on both the pre-recorded video input. The frames spikes up from the very beginning and level out evenly around the average of 2.5fps. It is also to be clarified that the data were collected from a mere 330 total number of frames, due to the inconsistent video duration from the inputs.

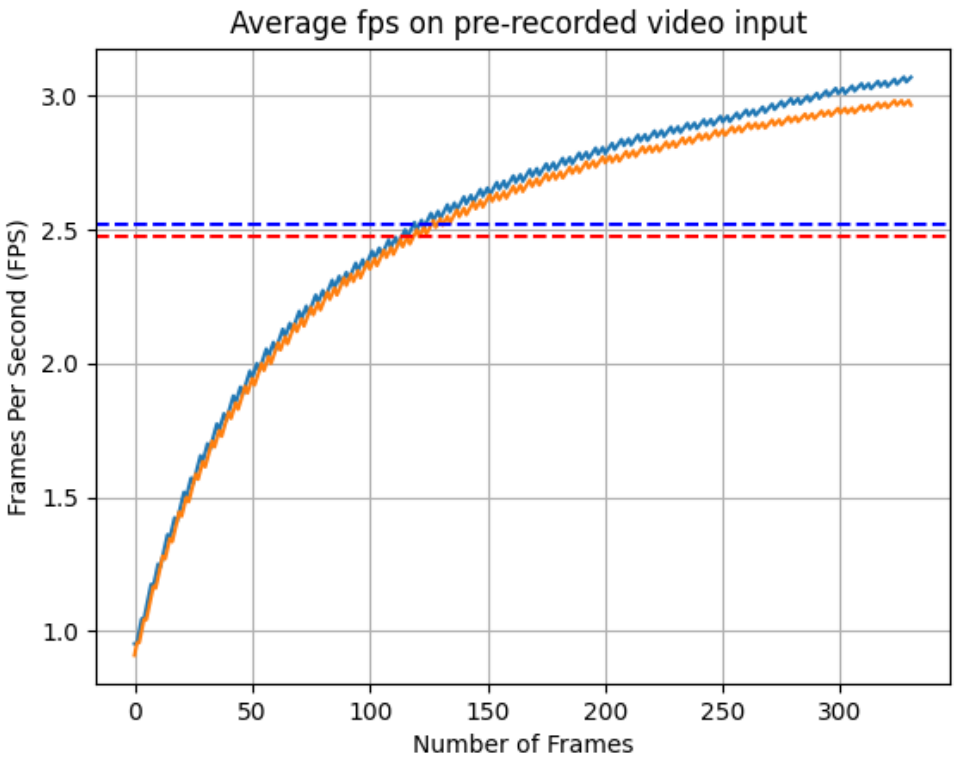


FIGURE 4.23 *– Average fps on pre-recorded video input from two distinct angles*

The receiver operating characteristic curve (ROC curve) is a graph that shows how well a classification model performs across all categorization levels. There are only two parameters are shown on this curve, true positive rate, and false positive rate. Hence, the ROC curve is simply the plot of the true positive rate (TPR) against the false positive rate (FPR). As the classification threshold is lowered, more items would be classified as positive resulting in an increase in both the false positives and true positives.

*The **True Positive Rate** can be defined by the following formula;*

$$TPR = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

*Whereas the **False Negative Rate** is defined as follows;*

$$FPR = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

The area under the curve (AUC) is a summary of the ROC curve that measures a classifier's ability to differentiate between classes. The AUC indicates how well the model differentiates between positive and negative classes. The greater the AUC, the better the model has done in predicting. For instance, for the case when AUC is 1, the classifier is capable of successfully distinguishing between all Positive and Negative class points. Contrarily, if the AUC is closer to 0, the classifier would classify all Negatives as Positives and all Positives as Negatives.

The precision recall curve (PRC), is used to monitor the efficiency and performance of binary classification systems. It is widely used in circumstances where the classes are severely disproportionate. The PRC plot provides a graphical depiction of the classifier’s performance over a range of thresholds, where the values of precision are plotted against recall. Ideally, the plot should have a decreasing value and if the average precision is 1, means that the system can correctly classify the true positive and negative cases. Average precision closer to zero indicates that a particular developed system does not perform well overall. Both the ROC and PR curves for the trained model in the first test, can be represented in the following figures; Figure 4.24 and 4.25.

Each batch has a set of 32 images which would be passed for training, which were then consecutively trained for 10 epochs. To be fair, these results were obtained on two random batches, rather than the whole test sample. On the first batch, the area under the curve, AUC was 0.81, whilst the average precision in the PR curve was at 0.81.

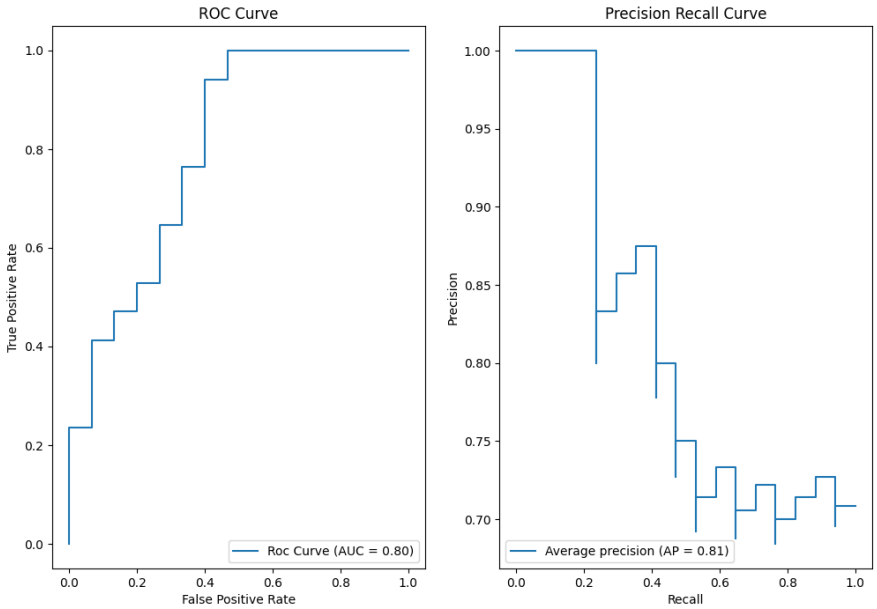


FIGURE 4.24 *– ROC Curve and Precision Recall curves from a pre-recorded video (1)*

On a second random batch, the area under the curve, AUC was 0.73, with an average precision in the PR curve of 0.64. This comes to show that this particular model, on average, does a decent job in identifying the person of interest against other people.

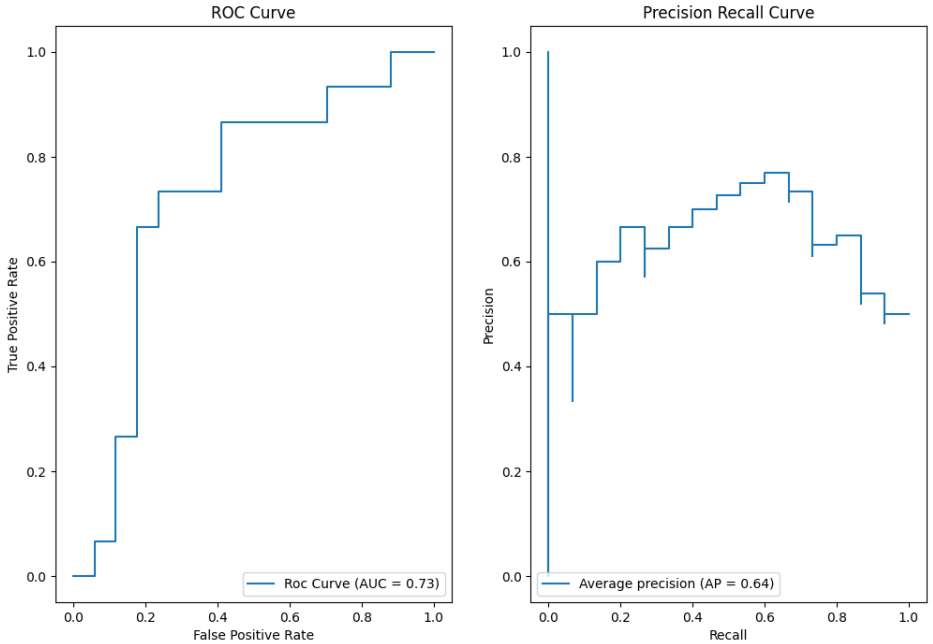


FIGURE 4.25 *– ROC Curve and Precision Recall curves from a pre-recorded video (2)*

The second test was performed by receiving a live stream input from the DCS5030L Internet Protocol camera. The figures 4.26, 4.27, 4.28 and 4.29 shows the test held in a bright room with minor occlusion. This time, the video input oversees an individual lingering across the frame, long enough to obtain enough images for training. Similarly, to collect images for training, the coordinates of the bounding boxes were used.

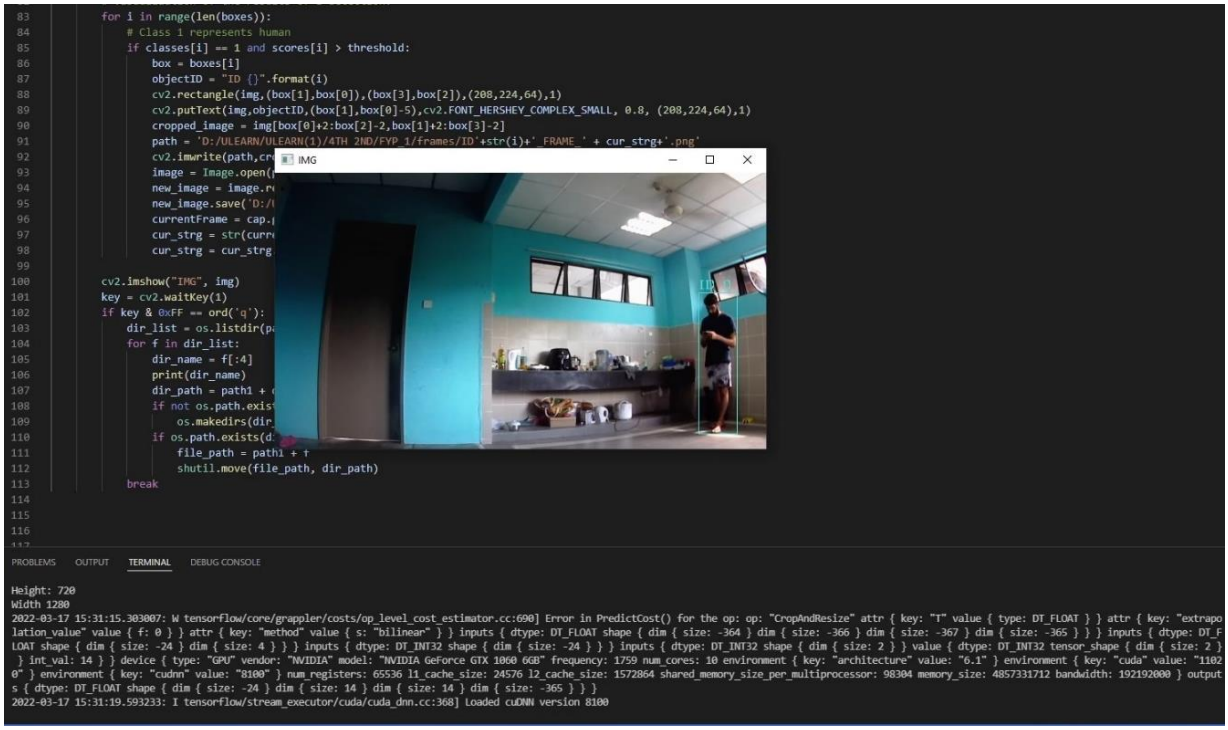


FIGURE 4.26 *– Video input from live stream HTTP (1)*

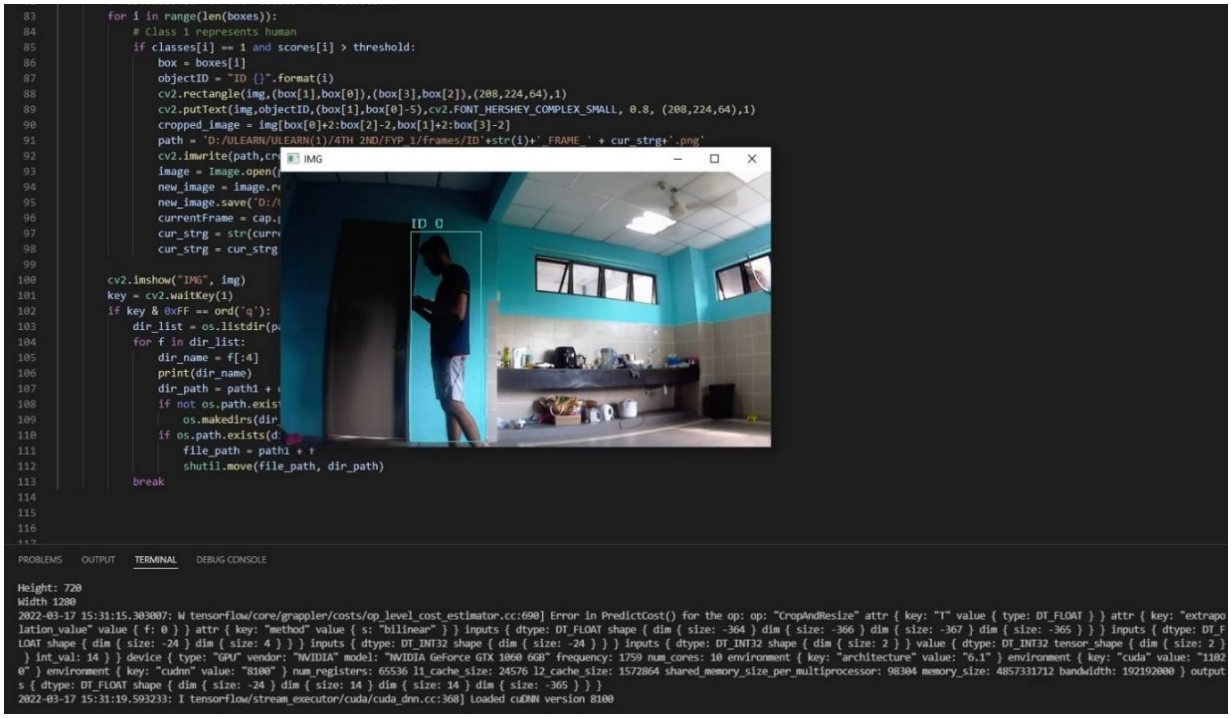


FIGURE 4.27 *– Video input from live stream HTTP (2)*

Since this was a test to challenge the accuracy of the predicted model, the individual in the video input purposely lingered longer than usual, at least if compared to a typical input from a live stream. This was to ensure that enough images were collected and resized before being trained into the network.

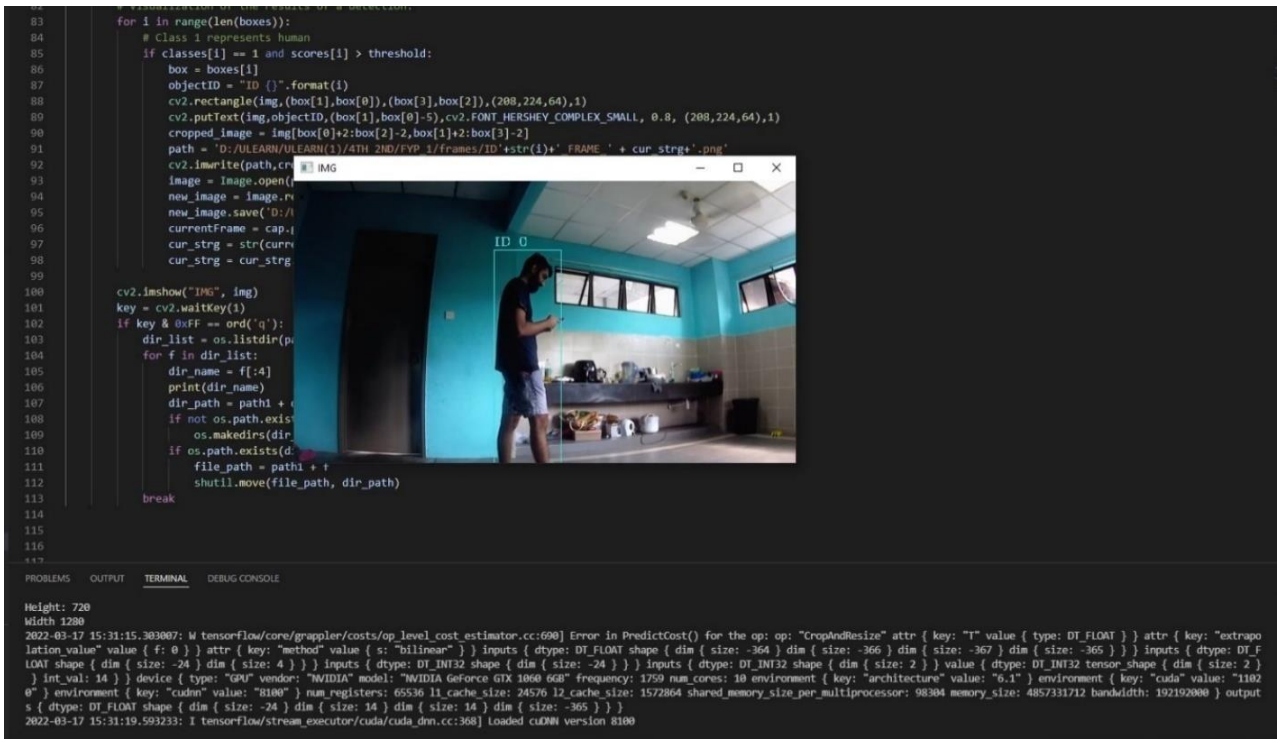


FIGURE 4.28 *– Video input from live stream HTTP (3)*

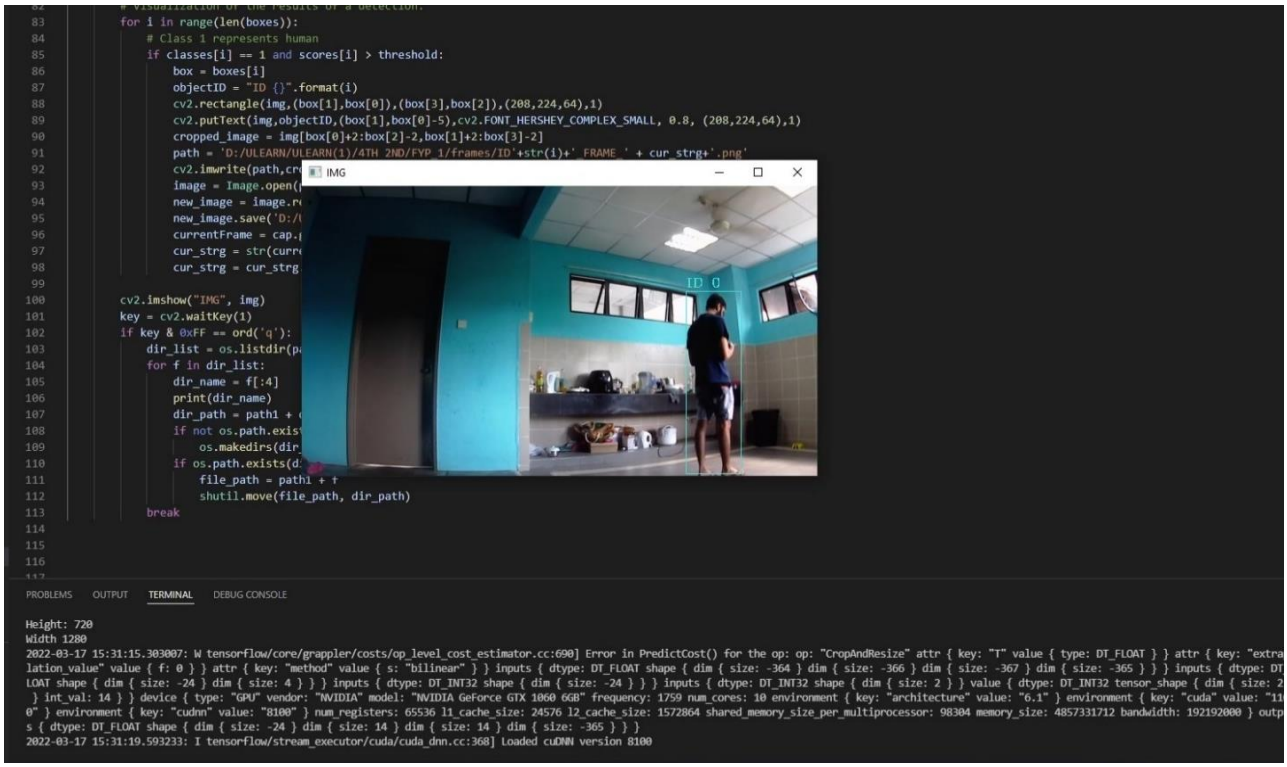


FIGURE 4.29 *– Video input from live stream HTTP (4)*

As mentioned previously, by having enough images the Siamese Model would be able to make predictions much more error-free and perfect by classifying that this person of interest is indeed the same person detected in a different camera, most likely from a different view. But first, the test was run on the camera from the same angle used to collect the train images. As can be seen in figures 4.30 and 4.31, the model detects the same person as **True**.

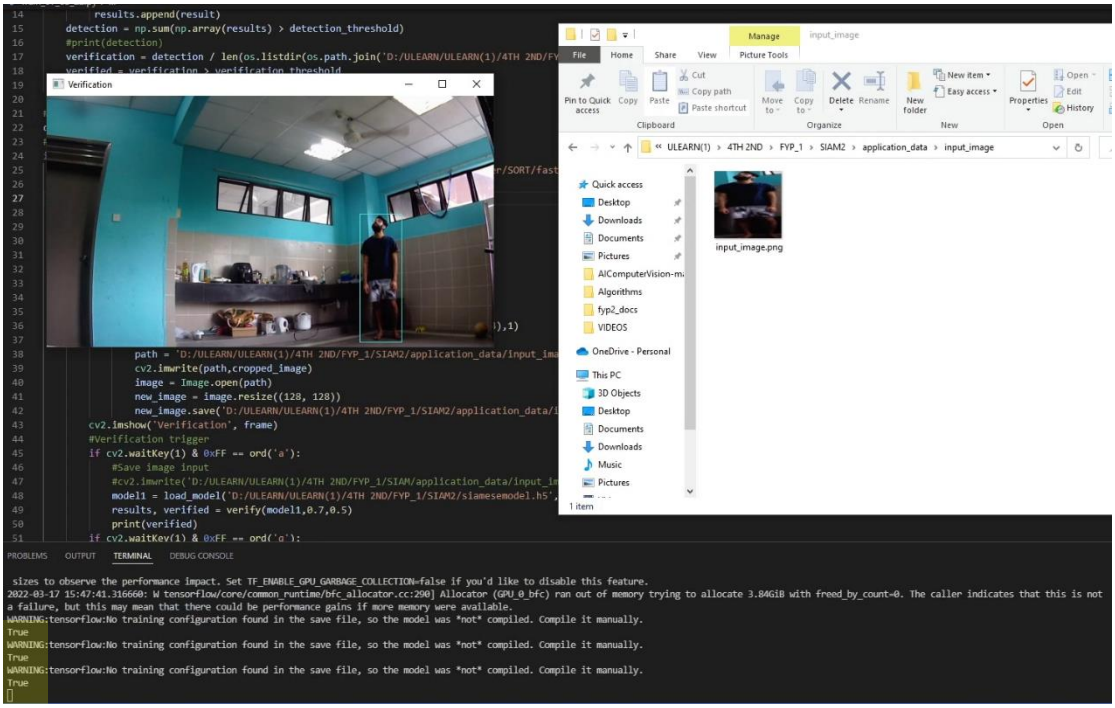


FIGURE 4.30 *– Video input from live stream, for testing (1)*

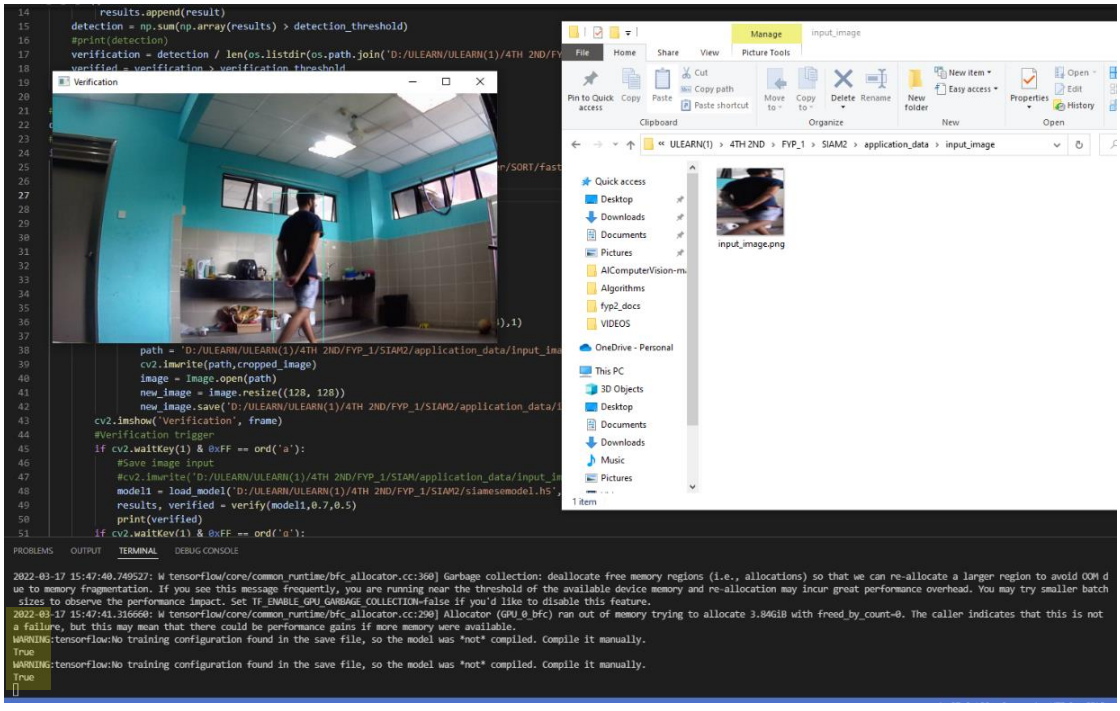


FIGURE 4.31 *– Video input from live stream, for testing (2)*

Now, the test was done by the same input, but from a different field of view. There are some false detections, but the overall performance is satisfactory. To tweak things up, the door was purposely let open to determine whether or not the Siamese Model trained can conclude that the person detected is the exact person used in training. The results were highlighted in yellow. Take a look in the figures 4.32 and 4.33 below.

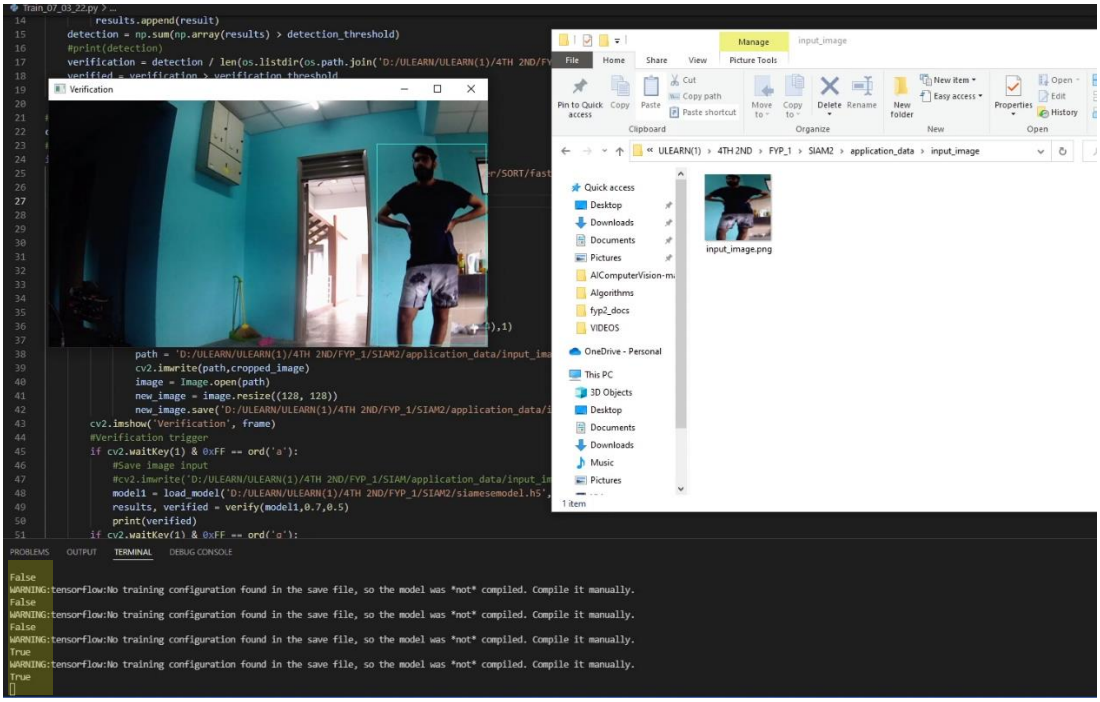


FIGURE 4.32 *– Video input from live stream, from a different angle (1)*

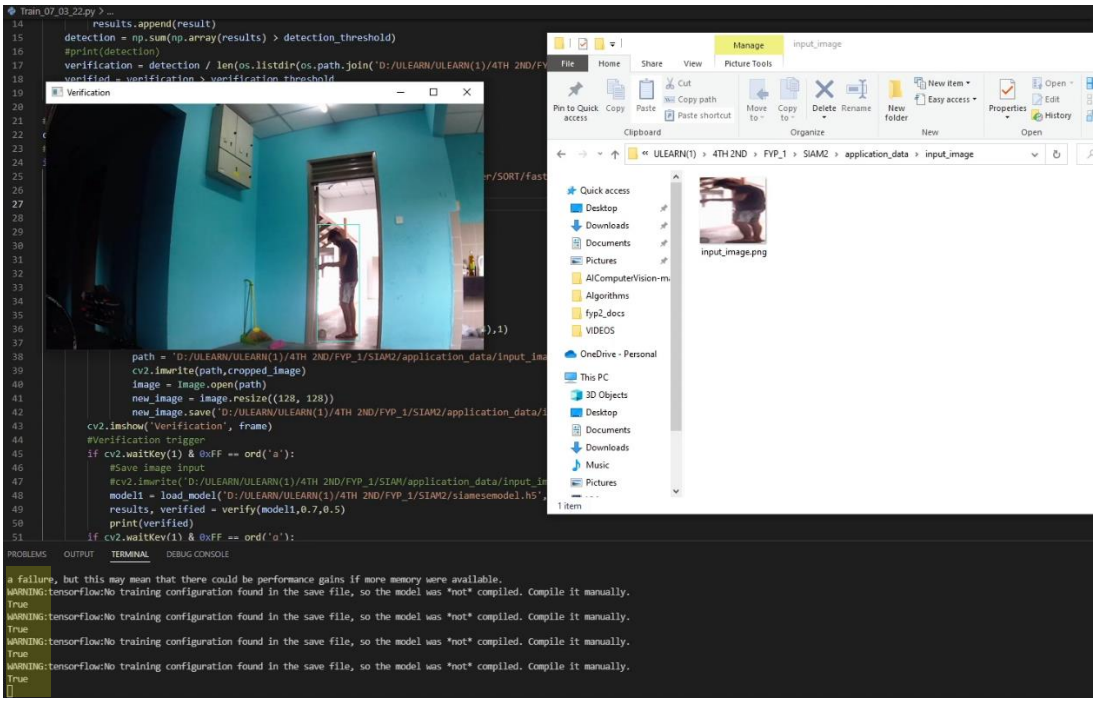


FIGURE 4.33 *– Video input from live stream, from a different angle (2)*

The average frames per second recorded from these two stream inputs were approximately, 2.08 and 2.569 respectively. Although, these results does not differ as much as the average frames per second recorded in the video input, there were some latency or delay that exist. This "*delay*" was not present in the pre-recorded video inputs. Hence it is safe to assume that this is due to the fact that the stream runs over the internet, accompanied by the use of hardware intensive object detection API and tracker that may have resulted this unprecedented delay. Since the project does not cover these aspects or issues, the root cause of the problem was left unexplored.

The figure 4.34 below represents the average fps on both the stream inputs. The average frames per second recorded was anywhere around 2.1 to 2.6fps. Similarly, the data were collected from 330 total number of frames, just to keep the measurements consistent for all scenarios. Since these were live streams from Hypertext Transfer Protocol (HTTP), some level of latency/delay was bound to happen, which may result in the decreased frame rate, if compared to an input from a pre-recorded video

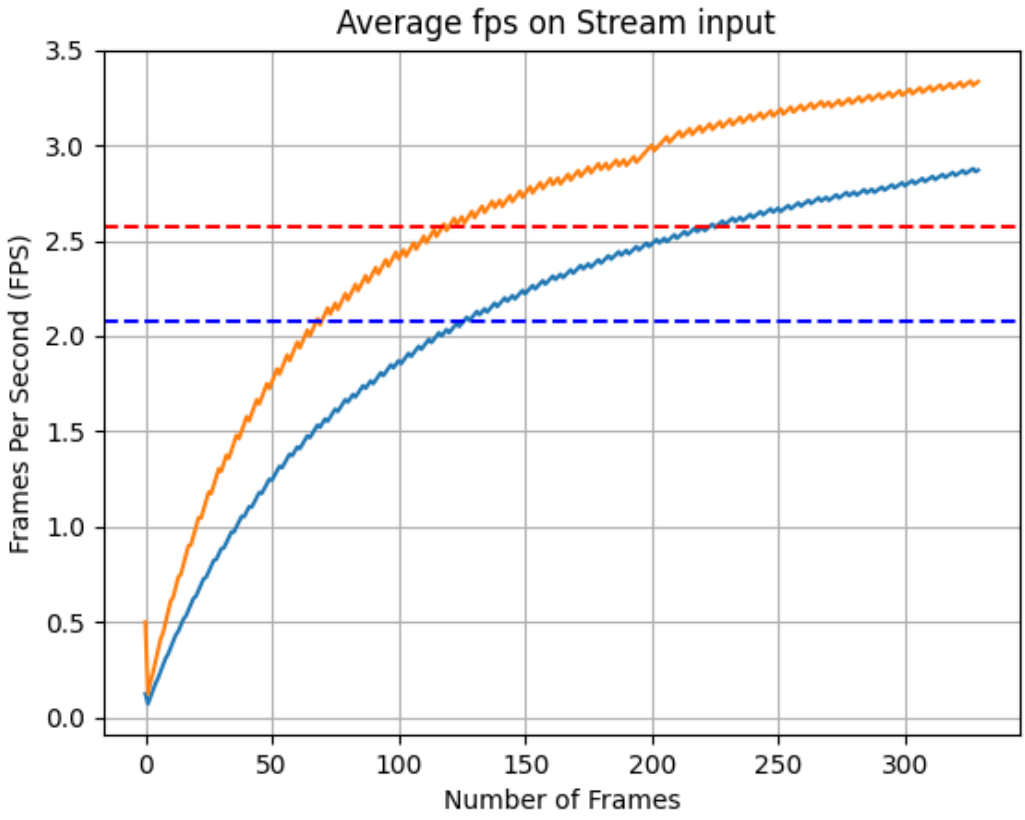


FIGURE 4.34 *– Average fps on stream input from two distinct angles*

Likewise, these results were obtained on two random batches, rather than the whole test sample. On the first batch, the area under the curve, AUC was 0.71, with an average precision in the PR curve of 0.64.

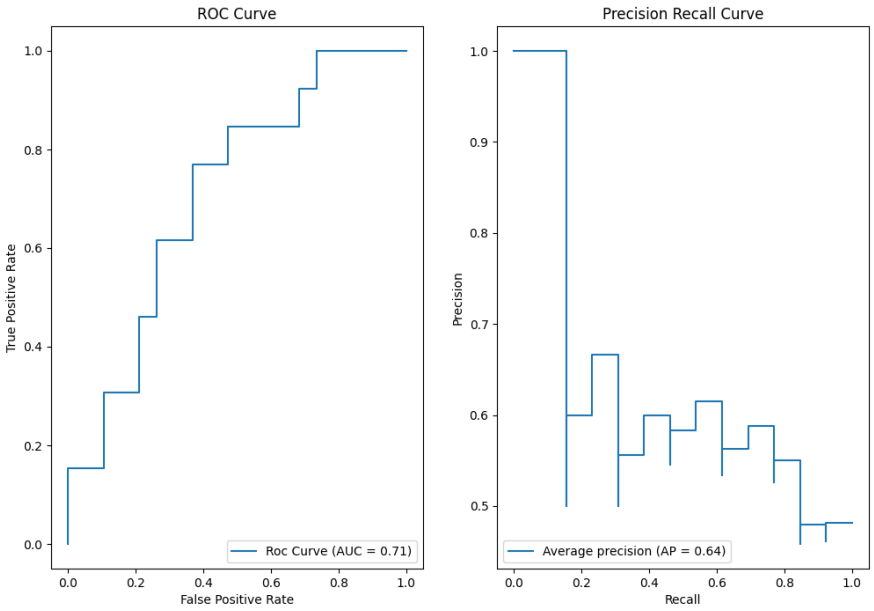


FIGURE 4.35 *– ROC Curve and Precision Recall curves from a stream (1)*

On the second random batch, the area under the curve, AUC was 0.60 with an average precision in the PR curve of 0.61. This model does a satisfactory job in identifying the person of interest, albeit not as accurate as the previously trained model. There could a few factors that could have resulted this; amount of light, occlusion, and the difference in the video resolution itself.

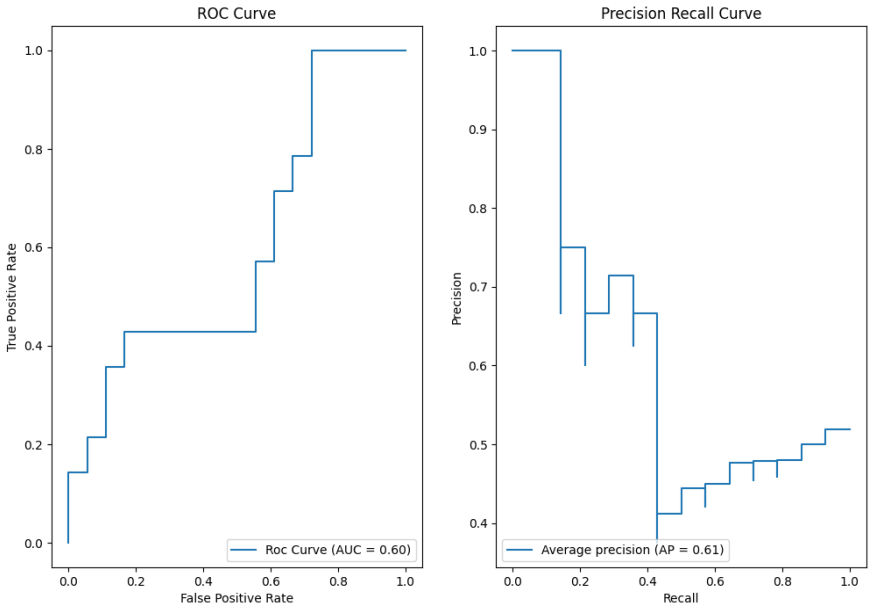


FIGURE 4.36 *– ROC Curve and Precision Recall curves from a stream (2)*

The third test was performed again by receiving a recorded video input from a phone camera. The figures 4.37 and 4.38 shows the test held in a bright open area with minor occlusion. Much similar to the second test, the third test was done along the hostel's corridor, accompanied with good lighting and ambience. This represents the first camera angle.

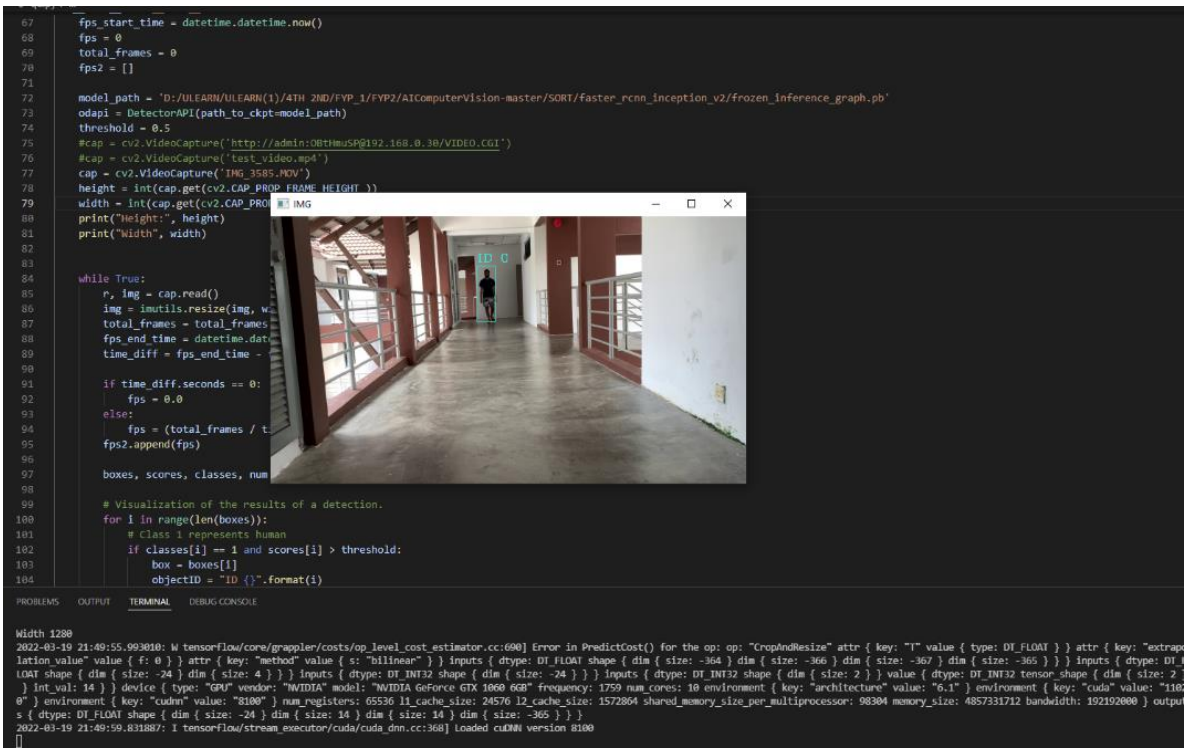


FIGURE 4.37 *–Input from recorded video (1)*

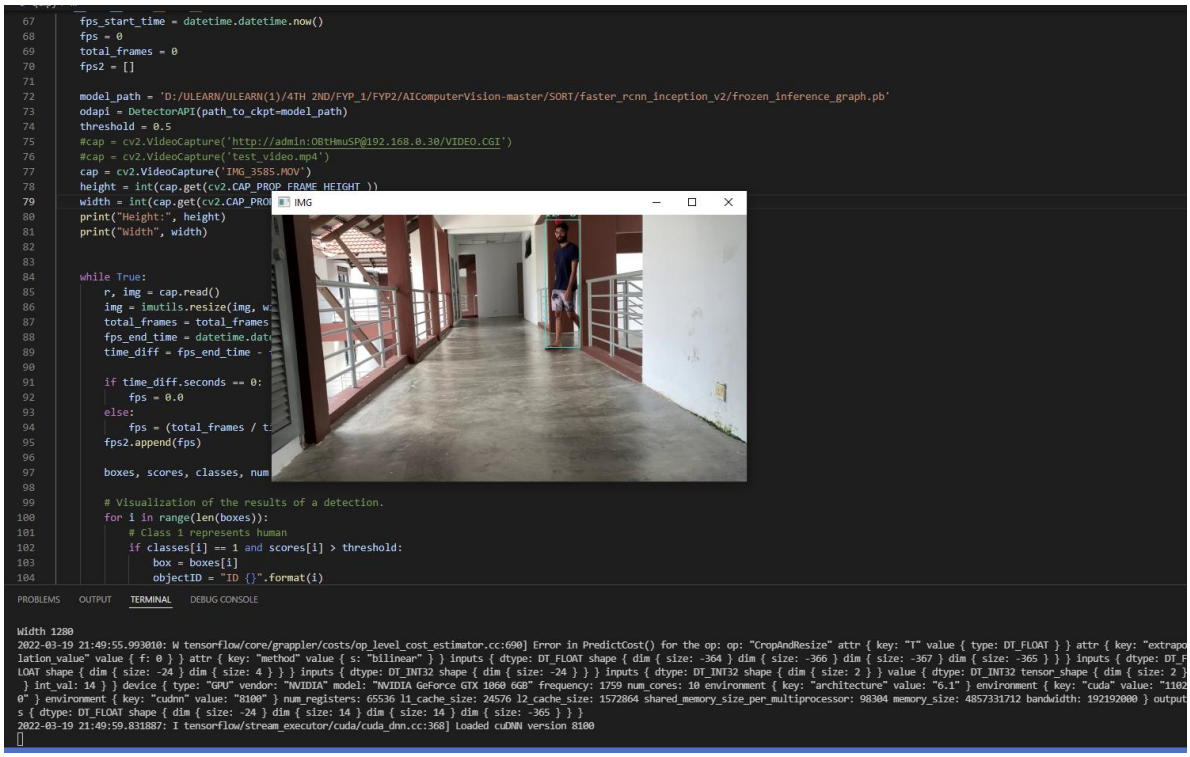


FIGURE 4.38 *– Input from recorded video (2)*

Now, since the video input was taken in the corridor (first camera angle), it would consecutively be used for training. Have a look at the figures 4.39, 4.40 and 4.41. The developed Siamese Model can identify the person of interest as the same person detected from the first video input, although this time the camera was placed on a different field of view.

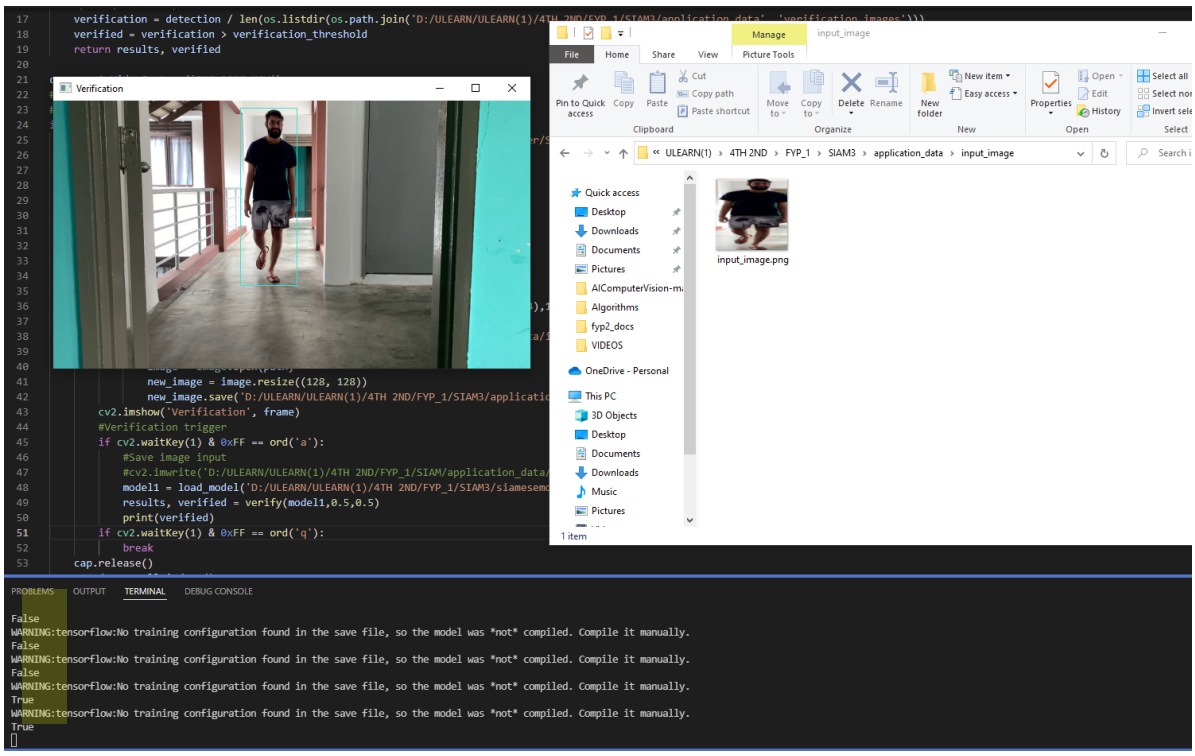


FIGURE 4.39 *–Input from recorded video, from a different angle (1)*

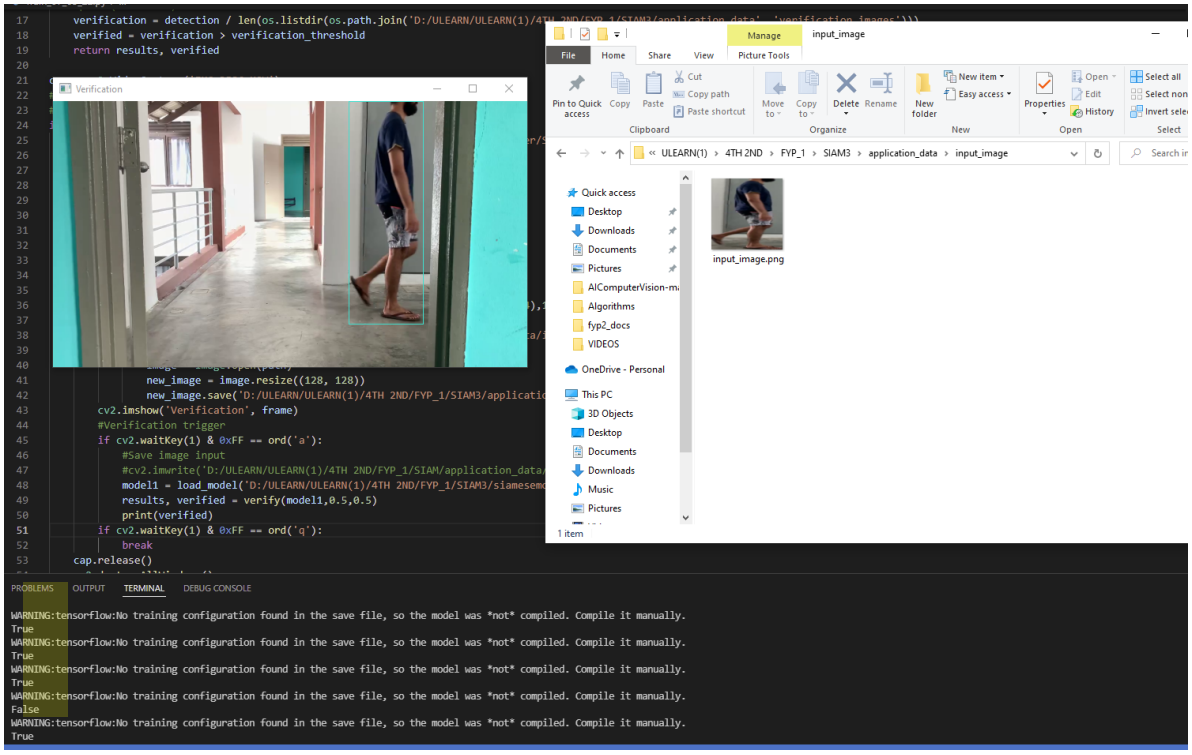


FIGURE 4.40 *– Input from recorded video, from a different angle (2)*

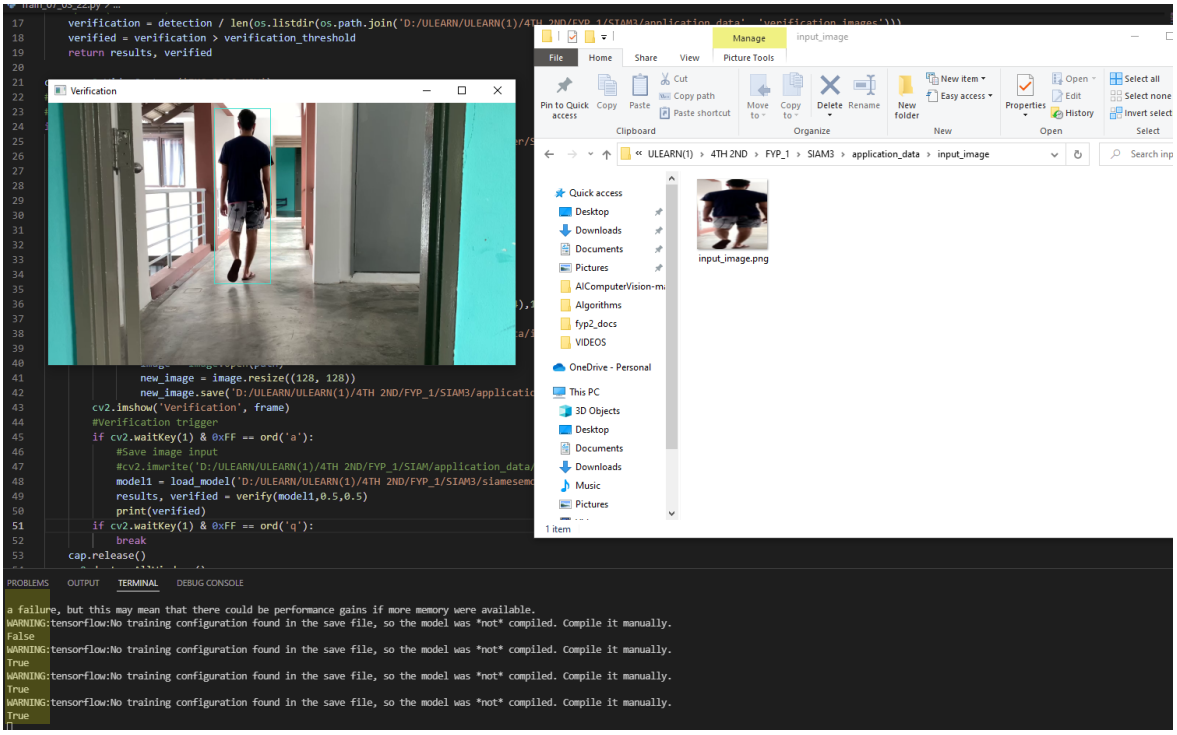


FIGURE 4.41 *– Input from recorded video, from a different angle (3)*

Generally, the Siamese model did a good job in predicting that this person, in the second video input is the same person detected in the first video input. In simpler words, the network was able to re-identify the person from a non-overlapping camera, that is of a different angle. Look at figures 4.42 and 4.43; was the same field of view, but with a different person present.

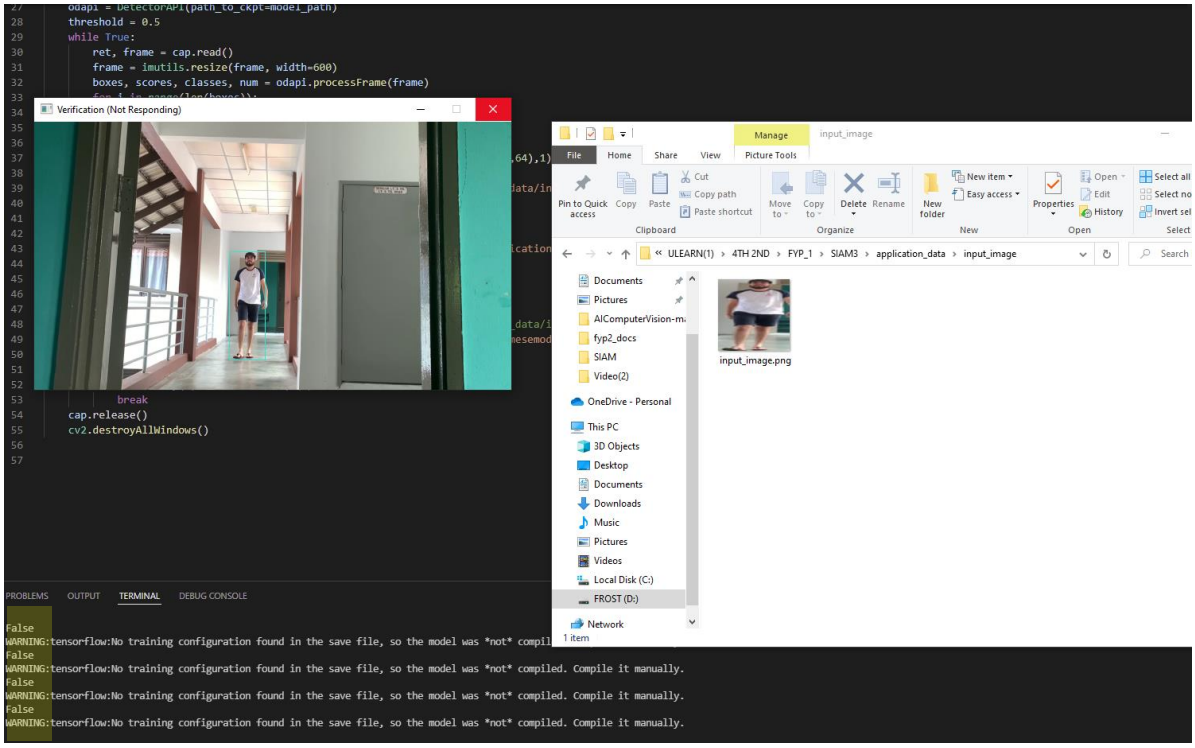


FIGURE 4.42 *– Input from recorded video, with a different person (1)*

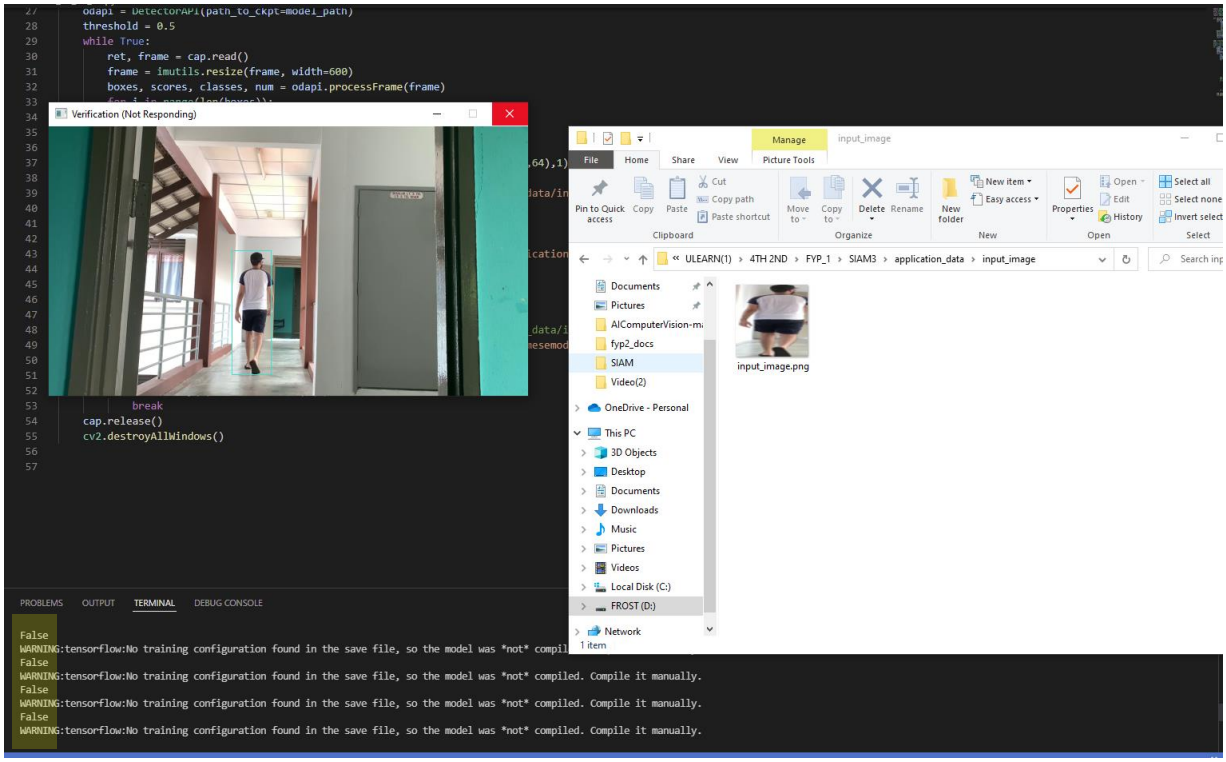


FIGURE 4.43 *– Input from recorded video, from a different angle (2)*

With a different person present, in the same camera angle, the Siamese model was able to make good predictions and concluded that this was not the same individual detected in the first camera. Now, have a look at figures 4.44, 4.45 and 4.46 instead.

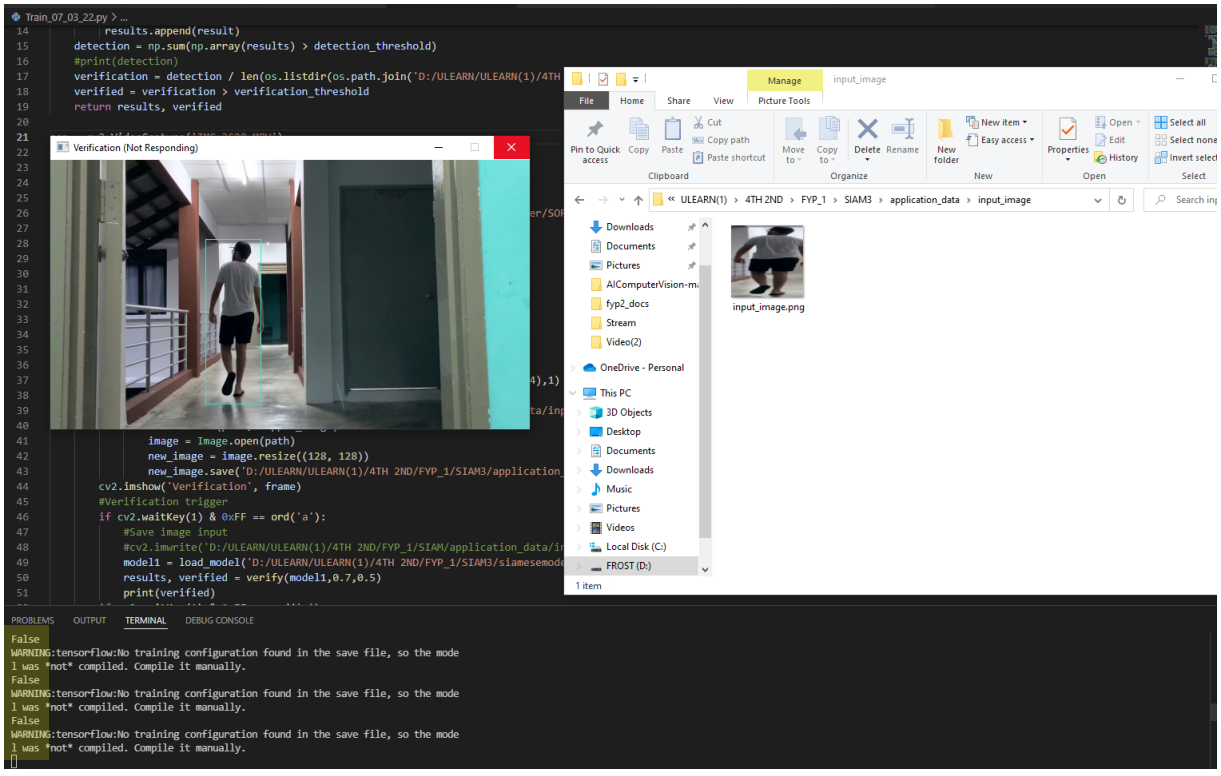


FIGURE 4.44 *– Input from recorded video, with a different lighting (1)*

To further question the legitimacy of the predicted model, the test was done on the same angle, with an unfamiliar person, once again but at a different lighting situation. The return results from the verification were **False**, indicating that this is not the person detected from the first camera input. Conclusively, the neural network has done an excellent job in reidentifying whether the individual tracked in the second camera angle, is the same person or not.

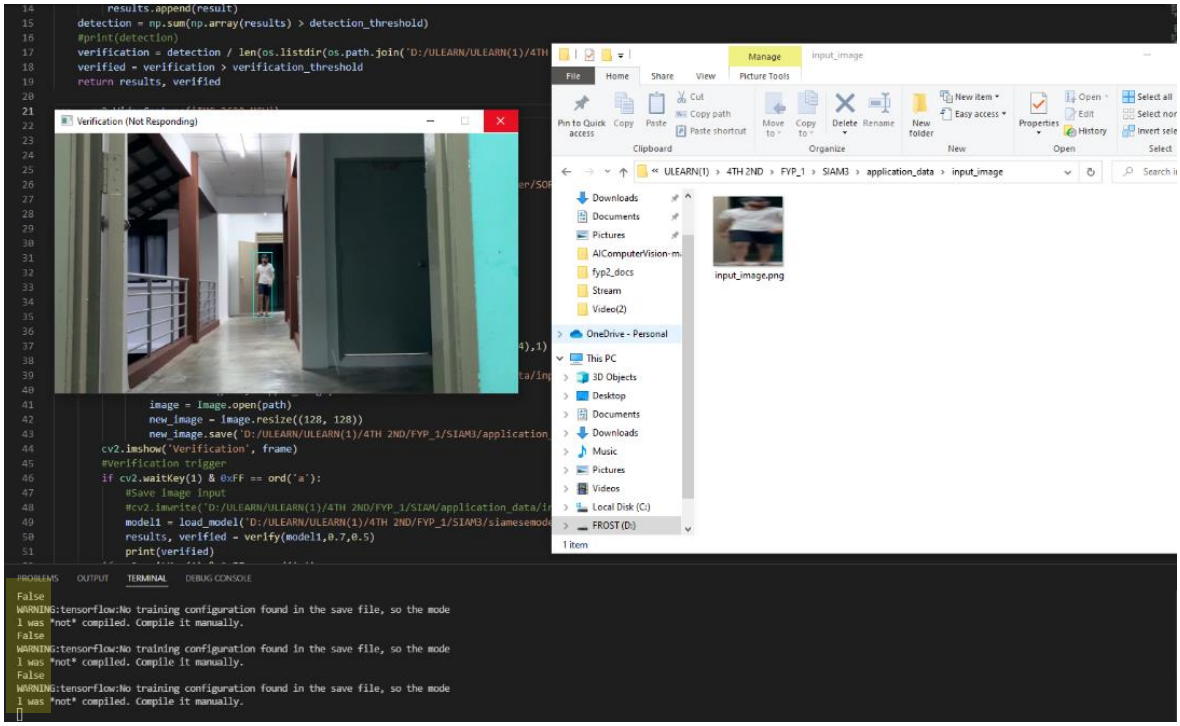


FIGURE 4.45 *– Input from recorded video, with a different lighting (2)*

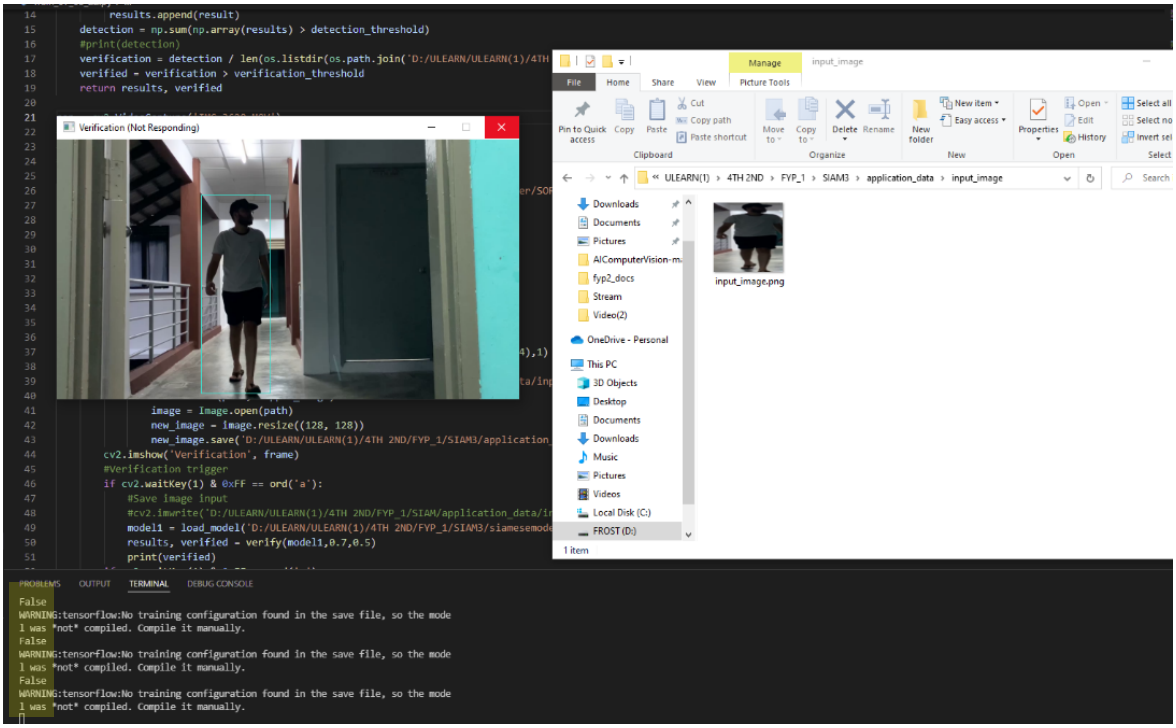


FIGURE 4.46 *– Input from recorded video, with a different person (3)*

The figure 4.45 below shows the average fps on all three of the pre-recorded video inputs. The frames spikes up from the very beginning and level out evenly around the average of 2.22, 2.98 and 2.50 fps respectively. As for all the other inputs, these data were also collected from a 330 total number of frames.

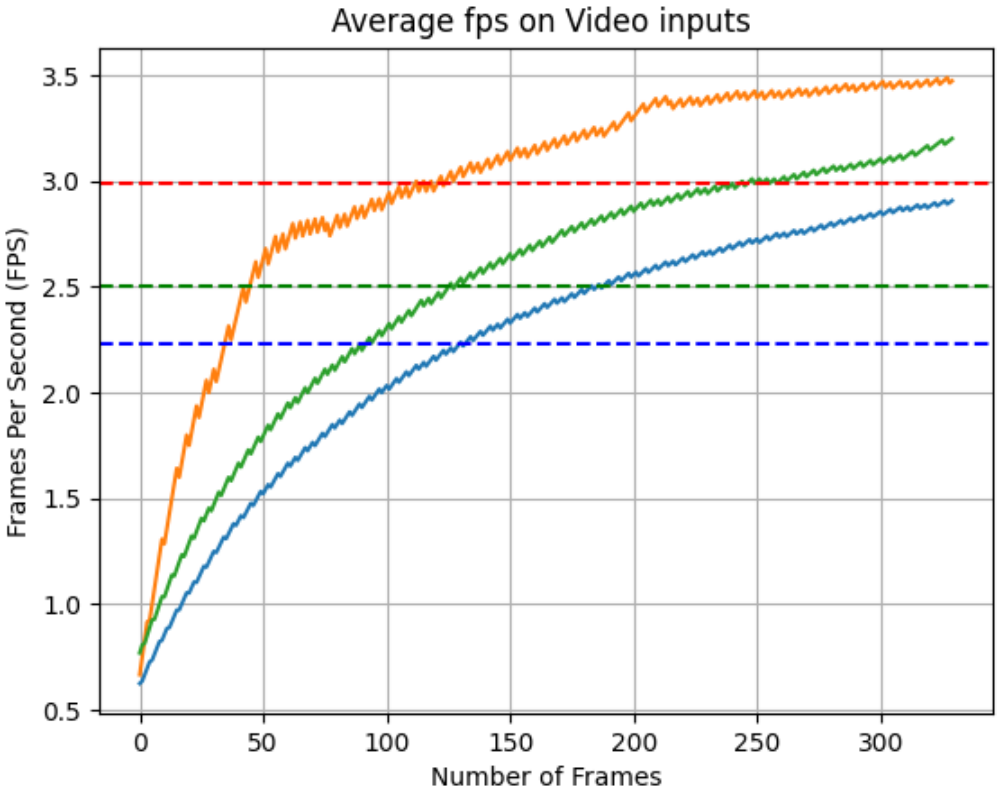


FIGURE 4.47 *- Average fps on pre-recorded video inputs from two distinct angles*

On the first random batch, AUC was 0.78, with an average precision in the PR curve of 0.80. This shows that the number of false positives and negatives were little.

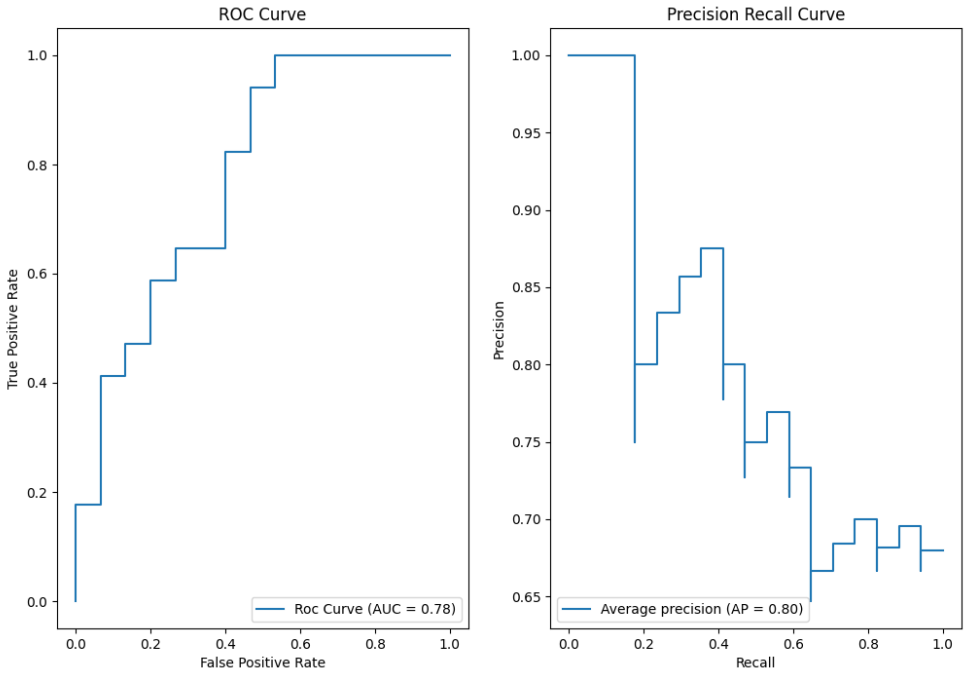


FIGURE 4.48 *– ROC Curve and Precision Recall curves from the pre-recorded videos (1)*

On a second random batch, the area under the curve, AUC was 0.68, with an average precision in the PR curve of 0.84. Similarly, this model also performed well in identifying the person of interest against other people.

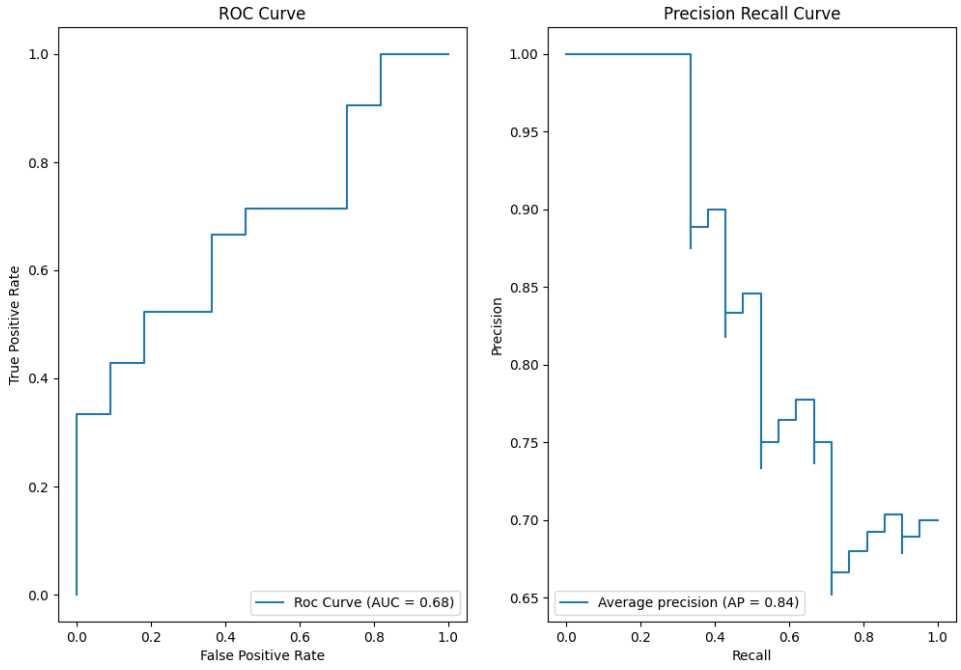


FIGURE 4.49 *– ROC Curve and Precision Recall curves from the pre-recorded videos (2)*

Since all the models take exactly 10 epochs for training, the difference between the time taken does not differ as much, regardless the type of input given. That being said, the average time taken to train the neural networks were approximately 190 to 310 seconds. As the number of epochs increases, the values of precision and recall, tends to get closer to 1, while loss, would get closer to zero. Loss is highlighted in yellow, green for precision and red for recall. Have a look at figure 4.50 below.

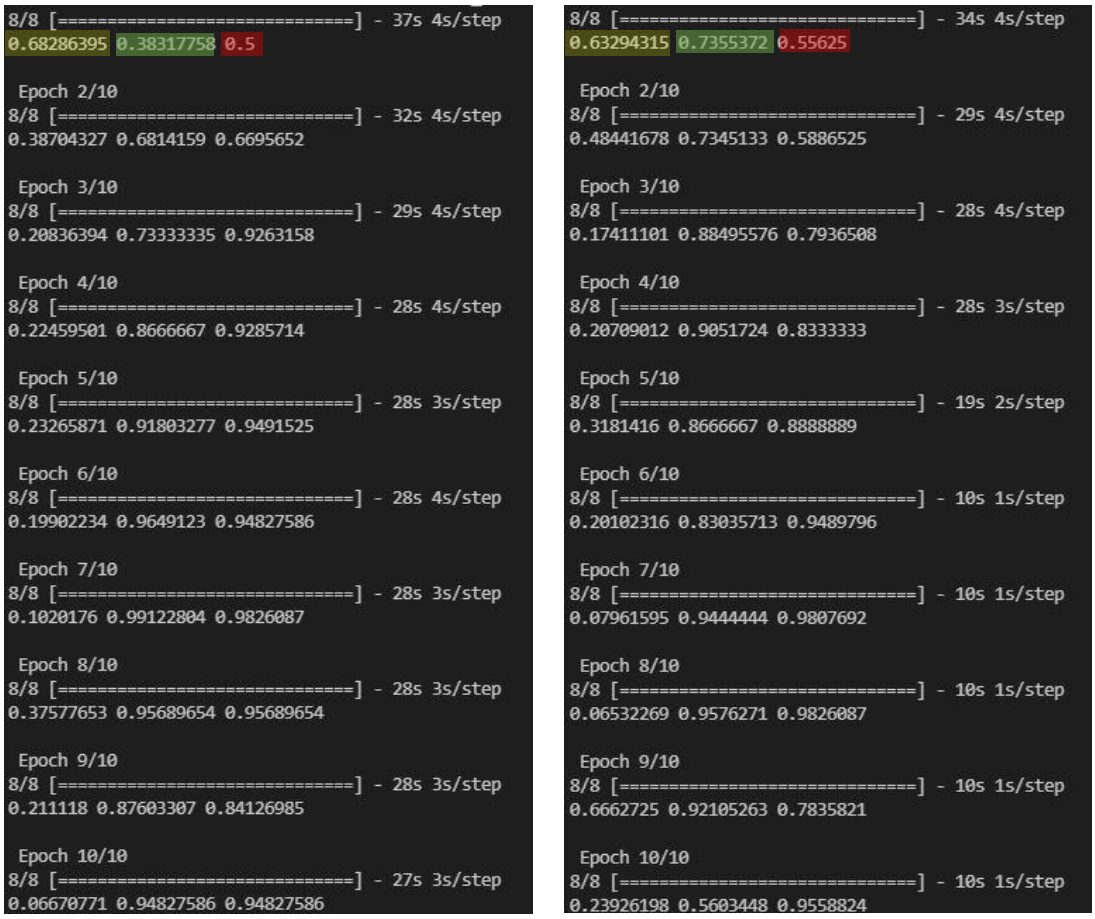


FIGURE 4.50 *- Average time taken to run 10 epochs*

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project represents a framework that implements a pedestrian re-identification network, which by itself may provide a wide range of advantages. With the urgent demand for public safety and an increasing number of surveillance cameras, the re-identification of people is of great importance. The development of such system is undeniably cost-effective; doesn't require the need of actual humans to do constant monitoring at all times. Hence, with the development of distributed video tracking the various methods for capturing parameters for the person of interest could be comprehended, and an existing network in a real-world scenario and on a new data can be verified and evaluated.

Thus far the project can only perform verification on a single person, from a given video input. This is due to the way the model was built upon, which lacks in the number of input images it can verify. Since the input image are detected by using the coordinates of the bounding box, the issue arrives when there more than a single person in a particular frame. In other words, during verification, the system cannot crop multiple images of multiple person in any given frame, to perform the verification. Hence, it is virtually impossible to point out the exact IDs of individuals, across two non-overlapping camera angles.

Hence, to further prevent complications as such, the whole project was built to ensure that only a single person is detected in the first, and second camera angles. This developed project would also work if there were multiple people in the frames as well, but instead of returning the IDs of the corresponding individuals, the system can only retrieve Boolean values of **True** or **False**. This means that, the neural network can identify whether or not all these people detected in the first camera angle, has reappeared in the second camera angle. However, it cannot distinguish the individuals of the already detected IDs and reassign them respectively.

Perhaps with more time and extensive testing, the system would be able to perform much better, by differentiating IDs which were assigned in the first camera, to simultaneously be detected in the second camera angle.

5.2 Recommendation

For developing such systems, its recommended to have the best in the line hardware, to be able to cop up with the extensive computation. This is especially useful if the training and deployment were to be done locally, and not over the internet. Contrarily, with NVIDIA Tesla T4 GPU with 16GB (gigabytes) of video memory on Google Colab would allow for quicker response time, and hence faster performance and higher frame rate. Besides, instead of using a streaming protocol like RTSP or HTTP, it is better to have cameras physically connected to a particular system to minimize overall delay.

It is also important to have footages recorded or streamed in a particularly bright area, with minor occlusion as it allows for far more excellent model training. The neural network design learns data point similarity using Siamese networks. Since the Siamese Neural Network make predictions based on images, it is best to ensure that the resolution and overall quality of an input video to be high, be it a live stream or from a recorded video.

CHAPTER 6: REFERENCES

[1] Klingler, N., 2021. *Deep Learning for Person Re-Identification - viso.ai*. [online] viso.ai. Available at: <https://viso.ai/deep-learning/deep-learning-for-person-re-identification/> [Accessed 1 October 2021].

[2] Sharabok, G., 2021. Why we need person re-identification?. [online] Medium. Available at: <https://towardsdatascience.com/why-we-need-person-re-identification-3a45d170098b> [Accessed 1 October 2021].

[3] PETER DANIELSON, M., 2021. *Monitoring and Surveillance | Encyclopedia.com*. [online] Encyclopedia.com. Available at: <https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/monitoring-and-surveillance> [Accessed 12 November 2021].

[4] A. McHale, Director, and Memoori Research, "2019 surpassed expectations but demand structure in video surveillance is unbalanced," *asmag.com*. [Online]. Available: https://www.asmag.com/showpost/30652.aspx. [Accessed: 12-Nov-2021].

[5] S. Feldstein, The global expansion of AI surveillance, vol. 17. Washington, DC: Carnegie Endowment for International Peace, 2019.

[6] J. Brownlee, "A gentle introduction to object recognition with deep learning," Machinelearningmastery.com, 21-May-2019. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/. [Accessed: 12-Nov-2021].

[7] L. Wu, C. Shen and A. van den Hengel, "PersonNet: Person Reidentification with Deep Convolutional Neural Networks", Computer Vision and Pattern Recognition, 2016

[8] L. Zhang, T. Xiang, and S. Gong, "Learning a discriminative null space for person re-I dentification," CVPR, 2016.

[9] F. Xiong, M. Gou, O. Camps and M. Sznaier, "Person Re-Identification Using Kernel-Based Metric Learning Methods", Computer Vision – ECCV 2014, pp. 1-16, 2014.

[10] S. Liao, Y. Hu, Xiangyu Zhu and S. Li, "Person re-identification by Local Maximal Occurrence representation and metric learning", 2015 IEEE International Conference on Signal and Image Processing Applications (IEEE ICSIPA 2019), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[11] 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, pp. 1363–1372

[12] E. H. Land and J. J. McCann, "Lightness and retinex theory," J. Opt. Soc. Am., vol. 61, no. 1, pp. 1–11, 1971.13)

[13] S. Salehian, P. Sebastian, and A. B. Sayuti, "Framework for pedestrian detection, tracking and re-identification in video surveillance system," in *2019 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2019.

[14] W. Li, R. Zhao, T. Xiao, and X. Wang, "DeepReID: Deep filter pairing neural network for person re-identification," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014..

[15] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, "FPNN: Field probing Neural Networks for 3D data," *arXiv [cs.CV]*, 2016

[16] "What is Object Tracking? - An Introduction (2021) - viso.ai," Viso.ai, 10-Jul-2021. [Online]. Available: https://viso.ai/deep-learning/object-tracking/. [Accessed: 12-Nov-2021].

[17] "Papers with code - object tracking," Paperswithcode.com. [Online]. Available: https://paperswithcode.com/task/object-tracking. [Accessed: 12-Nov-2021].

[18] A. Lukežič, T. Vojíř, L. Čehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *Int. J. Comput. Vis.*, vol. 126, no. 7, pp. 671–688, 2018.

[19] A. Yelisetty, "Understanding object detection and R-CNN," Towards Data Science, 08-Jul-2020. [Online]. Available: https://towardsdatascience.com/understanding-object-detection-and-r-cnn-e39c16f37600. [Accessed: 12-Nov-2021].

[20] "Object detection techniques in deep learning," Inblog.in. [Online]. Available: https://inblog.in/Object-Detection-Techniques-in-Deep-Learning-eTqP3Hneil. [Accessed: 12-Nov-2021].

[21] A. Yelisetty, "Understanding object detection and R-CNN," *Towards Data Science*, 08-Jul-2020. [Online]. Available: https://towardsdatascience.com/understanding-object-detection-and-r-cnn-e39c16f37600. [Accessed: 12-Nov-2021].

[22] R. Gandhi, "R-CNN, fast R-CNN, faster R-CNN, YOLO — object detection algorithms," *Towards Data Science*, 09-Jul-2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed: 12-Nov-2021].

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[25] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra and J. M. Z. Maningo, "Object Detection Using Convolutional Neural Networks," TENCON 2018 - 2018 IEEE Region 10 Conference, 2018, pp. 2023-2027.

[26] Kunar, A., 2021. Object Detection with SSD and MobileNet. [online] Medium. Available at: <https://adityakunar.medium.com/object-detection-with-ssd-and-mobilenet-aeedc5917ad0> [Accessed 26 October 2021].

[27] He, K., Gkioxari, G., Dollar, P. and Girshick, R., 2020. Mask R-CNN. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2), pp.386-397.

[28] J. F. Villacrés and F. Auat Cheein, "Detection and characterization of cherries: A deep learning usability case study in Chile," *Agronomy (Basel)*, vol. 10, no. 6, p. 835, 2020.

[29] A. Yelisetty, "Understanding object detection and R-CNN," Towards Data Science, 08-Jul-2020. [Online]. Available: https://towardsdatascience.com/understanding-object-detection-and-r-cnn-e39c16f37600. [Accessed: 12-Nov-2021].

[30] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, Cham: Springer International Publishing, 2016, pp. 21–37.

[31] Rosebrock, A., 2021. Simple object tracking with OpenCV - PyImageSearch. [online] PyImageSearch. Available at: <https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/> [Accessed 26 October 2021].

[32] Medium. 2021. People Tracking using Deep Learning. [online] Available at: <https://towardsdatascience.com/people-tracking-using-deep-learning-5c90d43774be> [Accessed 26 October 2021].

[33] Medium. 2021. People Tracking using Deep Learning. [online] Available at: <https://towardsdatascience.com/people-tracking-using-deep-learning-5c90d43774be> [Accessed 26 October 2021].

[34] AI & Machine Learning Blog. 2021. DeepSORT: Deep Learning to track custom objects in a video. [online] Available at: <https://nanonets.com/blog/object-tracking-deepsort/> [Accessed 26 October 2021].

[35] K. Host, M. Ivašić-Kos, and M. Pobar, "Tracking handball players with the DeepSORT algorithm," in *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods*, 2020.

[36] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu, "DCFNet: Discriminant Correlation Filters network for visual tracking," *arXiv [cs.CV]*, 2017.

[37] Liang, Feng & Yang, Shaofei & Mai, Tinrui & Yang, Yichen. (2018). The Design of Objects Bounding Boxes Non-Maximum Suppression and Visualization Module Based on FPGA. 1-5. 10.1109/ICDSP.2018.8631668.

[38] D. Wang *et al.*, "Daedalus: Breaking Non-Maximum Suppression in Object Detection via adversarial examples," *arXiv [cs.CV]*, 2019.

[39] L. M. Brown, R. Feris, and S. Pankanti, "Temporal non-maximum suppression for pedestrian detection using self-calibration," in *2014 22nd International Conference on Pattern Recognition*, 2014.

[40] G. Koch, R. Zemel, en R. Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition", 2015.

[41] N. I. Mokhtari, “What are Siamese neural networks in deep learning?,” Medium, 15-Feb-2022. [Online]. Available: https://towardsdatascience.com/what-are-siamese-neural-networks-in-deep-learning-bb092f749dcb. [Accessed: 08-Mar-2022].

[42] J. Brownlee, “A gentle introduction to the rectified linear unit (ReLU),” Machine Learning Mastery, 20-Aug-2020. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/#:~:text=The%20rectified%20linear%20activation%20function,otherwise%2C%20it%20will%20output%20zero. [Accessed: 08-Mar-2022].

[43] A. Bewley, Z. Ge, L. Ott, F. Ramos, en B. Upcroft, “Simple online and realtime tracking”, in 2016 IEEE International Conference on Image Processing (ICIP), 2016, bll 3464–3468.

[44] N. Wojke, A. Bewley, en D. Paulus, “Simple Online and Realtime Tracking with a Deep Association Metric”, in 2017 IEEE International Conference on Image Processing (ICIP), 2017, bll 3645–3649.

[45] N. Wojke en A. Bewley, “Deep Cosine Metric Learning for Person Re-identification”, in 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, bll 748–756.