

Ministry of Education and Science of Ukraine
Ternopil Ivan Puluj National Technical University

Faculty of Computer Information System and Software Engineering

(full name of faculty)

Department of Computer Science

(full name of department)

QUALIFYING PAPER

For the degree of

Bachelor

(degree name)

topic: **Design of Biometric Gait Recognition System**

Submitted by: fourth year student _____, group ICH-42
specialty 122 Computer Science

(code and name of specialty)

	_____	Saleha Khizar
	(signature)	(surname and initials)
Supervisor	_____	N. Zagrodna
	(signature)	(surname and initials)
Standards verified by	_____	_____
	(signature)	(surname and initials)
Head of Department	_____	I. Bodnarchuk
	(signature)	(surname and initials)
Reviewer	_____	_____
	(signature)	(surname and initials)

Ministry of Education and Science of Ukraine
Ternopil Ivan Puluj National Technical University

Faculty Faculty of Computer Information System and Software Engineering
(full name of faculty)

Department Department of Computer Science
(full name of department)

APPROVED BY

Head of Department

I. Bodnarchuk

(signature)

(surname and initials)

<< >>

20__

ASSIGNMENT
for QUALIFYING PAPER

for the degree of Bachelor
(degree name)

specialty 122 Computer science
(code and name of the specialty)

student Saleha Khizar
(surname, name, patronymic)

1. Paper topic Design of Biometric Gait Recognition System

Paper supervisor Nataliya Zagorodna
(surname, name, patronymic, scientific degree, academic rank)

Approved by university order as of << 17 >> 12 2021 № 4/7-1068

2. Student's paper submission deadline 1th-July-2022

3. Initial data for the paper _____

4. Paper contents (list of issues to be developed)

CHAPTER 1. LITERATURE REVIEW

CHAPTER 2. DESIGN OF SYSTEM OF GAIT RECOGNITION

CHAPTER 3 IMPLEMENTATION AND TESTING

CHAPTER 4. LABOUR PROTECTION AND SAFETY IN EMERGENCY

5. List of graphic material (with exact number of required drawings, slides)

6. Advisors of paper chapters

Chapter	Advisor's surname, initials and position	Signature, date	
		assignment was given by	assignment was received by
LABOUR PROTECTION AND SAFETY IN EMERGENCY			

7. Date of receiving the assignment

TIME SCHEDULE

LN	Paper stages	Paper stages deadlines	Notes
	Literature Review		completed
	Chapter 1		completed
	Chapter 2		completed
	Chapter 3		completed
	Chapter 4		completed
	Chapter 5		completed
	Sub System Architecture		completed
	Pre-processing Algorithm		completed
	Research for GEI's		completed
	Graphical User Interface		completed
	Labor protection and safety in emergency		completed
	Implementation of ResNet		completed
	Plagiarism Check		
	Defense of qualifying paper		

Student

_____ (signature)

Saleha Khizar

_____ (surname and initials)

Paper supervisor

_____ (signature)

N.Zagorodna

_____ (surname and initials)

ABSTRACT

Design of Biometric Gait Recognition System // Thesis of educational level "Bachelor" // Saleha Khizar// Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group ICH-42 // Ternopil, 2022// p.-___, fig.-___,table-___, drawing-___,appendices-___.

Key words: GAIT RECOGNITION, ANALYSIS OF EXISTING GAIT SYSTEMS, DATA ACQUISITION, PRE-PROCESSING, FEATURE EXTRACTION, CLASSIFICATION, ARCHITECTURE DESIGN, SUSYSTEM ARCHITECTURE, PRE-PROCESSING ALGORITHM IMPLIMENTATION, GAIT ENERGY IMAGE, GRAPHICAL USER INTERFACE, CASIA-B, RESNET34.

In this era of innovation and technology, new things are being introduced daily in every field. Recently, a new biometric named gait is being researched a lot to develop it into biometric authentication and recognition system for intelligent surveillance. Gait is defined as a person's way of walking and is unique to every person. Gait has various attractive properties making it superior to other biometrics.

The aim of the project is to design a new gait recognition system based on one of the latest deep learning residuals convolutional neural network ResNet34. The results of applying ResNet34 to publicly available datasets CASIA-B clearly indicates that this model shows promise in this area. To develop a completely functional product using the proposed model, I first collected gait data of an individual by making them walk in front of camera several times, for this I took the online video as I was not able to go to the university and use pedestrian camera. Background subtraction is then applied to the video frames to extract silhouettes and various morphological operations to remove noise and shadows. Gait Energy Image is used as feature extraction technique and finally a pretrained ResNet34 model is used to perform identification at real-time for this I used

the CASIA-B samples as collecting my own data was not possible. Comparing the obtained results with other state-of-the-art models, my model achieves highest accuracy in more than half of the cases. This indicates that ResNet34 can achieve ground-breaking results in gait recognition field. In future, using better pretreatment and feature extraction techniques, I strongly believe that ResNet34 can be the next state-of-the-art model in gait recognition.

TABLE OF CONTENTS

INTRODUCTION.....	8
CHAPTER 1. LITERATURE REVIEW	12
1.1 History	12
1.2 Previous Solutions.....	13
1.3 Gait Databases.....	14
1.4 Problem Definition.....	15
1.5 Methodology	16
1.5.1 Methodological Approach.....	16
1.5.2 Methods of Data Collection/Selection	16
1.5.3 Methods of Analysis.....	18
CHAPTER 2. DESIGN OF SYSTEM OF GAIT RECOGNITION.....	19
2.1 Mathematical Fundamentals of System Design	19
2.1.1 Data Acquisition.....	19
2.1.2 Pre-Processing	21
2.1.3 Feature Extraction	23
2.1.4 Classification	24
2.2 System Architecture	25
2.2.1 Architecture Design Approach.....	25
2.2.2 Architecture Design.....	26
2.1.3 Subsystem Architecture.....	26
2.3 Detailed System Design.....	28

2.3.1 CCTV Camera.....	28
2.3.2 Pedestrian Detection.....	29
CHAPTER 3 IMPLEMENTATION AND TESTING	35
3.1 Pre-Processing Algorithm Implementation	35
3.2 Gait Energy Image (GEI).....	42
3.3 Graphical User Interface (GUI)	44
3.4 Class Diagram	47
3.5 Training And Testing Resnet	48
3.6 Implementation Of Resnet.....	49
3.7 Gallery Probe Testing.....	52
3.8 Results and Discussion.....	53
CHAPTER 4. LABOUR PROTECTION AND SAFETY IN EMERGENCY	55
4.1 Labour Protection.....	55
4.2 Emergency safety	59
CONCLUSION	62
REFERENCES.....	63

INTRODUCTION

With the dawn of technology, everything is being computerized from personal information of a person to confidential secrets of agencies. Each individual is storing all of their data on digital systems because as humans we always look for ease of access. Due to this, security of these digital systems has become one of the major concerns. Previously, people used traditional security measures such as pin, patterns, passwords etc. but as computers are becoming more and more advanced, these traditional measures are becoming easy to crack. In the past few decades, new techniques have been developed which provides security based on the unique features of an individual such as fingerprint scanners, facial recognition, retinal scans, gait recognition. Since these techniques are based on behavioral or physiological or biological characteristics of an individual, they are termed as biometrics. Example of such biometrics include fingerprints, retina, face, finger veins, palm veins, voice etc. With these biometric systems, it becomes difficult to break into the system. But recently a physiological characteristic of an individual is being developed into a new biometric which is called gait.

Gait is defined as an individual's way of walking or coordinated, cyclic movement of limbs that produce human locomotion. So, gait recognition is identifying a person through their manner of walking. Every person has an inimitable muscular-skeletal structure while walking, thus making gait recognition feasible and a possibility.

Example of movements that are gaits include walking, running, jogging, and climbing. Movements such as sitting or standing up from a chair are not considered gait because they are not cyclic movements. Moreover, jumping also cannot be considered as gait because it does not involve locomotion.

Just like every other biometric feature, it has been proved biologically that gait of every individual is unique to that person. It may seem that the two persons are walking in the same manner, but if you capture their walks and compare them on a computer, you will be amazed to see the differences between the two.

People believe that they can identify a familiar person from a far-off distance just by looking at their gaits. Psychology also suggests that human gait has 24 unique components which can be used to identify them. These experiences along with the development in machine learning and computer vision has led to the use of gait recognition as biometric tool for authentication.

Although there are other biometric systems which are in abundant use today, gait recognition systems have certain attractive features and properties making it a more promising system than others.

i) Can work remotely, that is human gait can be recognized from a far-off distance as compared to other biometric systems which require close distances or contact with the system. For example: retina scan or facial recognition requires the individual to be as close to the camera as possible. Similarly, fingerprint scanner requires the individual to put his/her finger on the device.

ii) Can even work quite accurately when other factors are hidden such as when face of an individual is hidden. However, hiding human motions is difficult. A person walking in a certain direction simply cannot hide the movement of his arms or legs for instance.

iii) Can be done without the cooperation of individuals. E.g., fingerprint scanners require individual to put his/her fingers on the device. On the other hand, gait recognition does not require any form of user-system interaction.

iv) Can be implemented with simple instrumentation such as a CCTV camera having a low resolution. Whereas other biometric systems require high quality images e.g., facial recognition or retinal scans.

v) Based on human silhouettes and motions which are hard to be impersonated.

vi) Can be used with other biometric systems to provide two-factor authentication.

Gait recognition system proposed in this report has a huge market. It can be used everywhere where another biometric system is being used such as fingerprint scanner or

facial recognition and it can be implemented in a more cost-effective manner as compared to other biometric systems.

The biggest use of gait recognition system is by law enforcements agencies. These agencies have been using other biometrics in criminal investigation for years. With a new biometric added, it would increase the size of database and thus aid in investigations.

Another main area of gait recognition systems is at the airports. For immigration control, the main biometric being used all over the world is iris recognition for which passengers have to wait in line for many hours. This can easily be replaced with gait recognition. Since passengers already have to walk from airplane to the exit of airport, a simple walking booth can be installed in-between with white backgrounds and cameras installed at different places which record the gait of every individual at different angles and stores it inside the database. In this way, the long waiting time of iris recognition can be avoided.

One more example includes Universities as this system will help each and every student to be recognized at the main entry point and help the University system to mark their attendance accordingly and obtain that data more accurately, this will save time and professors energy during classes as all of the data would already be registered in the system when they first entered the campus.

And last but not the least, gait recognition systems can be installed where there is a single point entry of authorized users only such as inside labs or high security areas.

The rest of the document is organized as follows. Chapter 1 talks about the previous researches that has been done on gait analysis, various solutions proposed and also different gait datasets available for the public for research purposes. This chapter also highlights the problem which I am tackling through my proposed system. Second chapter specifies the various data collection and data selection techniques which I used along with the approach used for doing the research part. The chapter also enlists the subsystems my model is divided into and also presents the in-depth details of these subsystems along with various modules and packages these subsystems are divided into. The frontend of my final product was research part as I required the GPU and I was not in Ukraine due to

unfortunate war situation so provided with class and ER diagrams at the end of this chapter and how it could be built. Third Chapter mentions the research of implementation details and how the datasets are divided to perform the final testing. It also tabulates the final results obtained and finally this chapter concludes with final remarks and some direction for the future work in this field as well as for my proposed solution.

CHAPTER 1. LITERATURE REVIEW

1.1 History

Gait recognition has attracted attentions of many researches in the past two decades. The first gait analysis system was proposed in 1994 by Niyogi and Edward H. Adelson[1]. First ever database consisting of gait data was set up in 2005 by Defense Advanced Research Projects Agency (DARPA)[2] and was made available for general public after which work on gait recognition systems paced up.

Early gait recognition systems used video-based data. Two approaches are used in gait analysis based on video data, model-based and model-free.

Model-based approach makes a model of the human body and then extracts features from the proposed model. The first model-based gait recognition system was proposed in 1997 by David Cunado, Mark S. Nixon and John N. Carter[3] in which they used pendulum for modeling the movement of legs which was then used for gait recognition. In 2003, first 2D-figures of human body were used in gait recognition by Jang-Hee Yoo[4]. In 2004, first 3D-figures of human body were modeled in gait recognition systems by Urtasun and Fua[5].

Model-free approaches focuses more on the silhouettes or the whole motion of human bodies instead of human body models. First model-free gait recognition system was proposed by Han and Bhanu[6] in 2006 which represented features in the video as a single image. This system was named Gait Energy Image (GEI). Another model-free gait recognition system was proposed in 2007 by Liu and Zheng[7], it was named Gait History Image(GHI).

In the last two decades, with the introduction of sensors, various of these sensors have been used to collect gait data as well such as accelerometer, floor sensors and continuous wave radars. First accelerometer-based gait system was introduced in 2005 by Ailisto and Makela[8]. Otero[9] in 2005 introduced the first continuous wave radar based

gait system. Floor sensors were used in gait system for the first time by Nakajima[10] in 2000.

Table 1.1- Gait recognition system milestones

Year	Main Idea
1994	First ever gait recognition system
1997	First model-based gait-recognition system.
2005	First accelerometer-based gait recognition system.
2006	First model-free gait recognition system based on Gait Energy Image GEI.
2007	Model-free gait recognition system based on Gait History Image GHI.

1.2 Previous Solutions

With publicly available gait databases, a lot of gait recognition systems has been proposed in theory, each better than the one before it. In the last decade, advancement in the computer vision and deep learning has made gait recognition a relatively easy task. The main problem with gait systems was large amount of data since they operate on video files. New deep learning models has made the processing of large amounts of data faster and computer vision has made it easier to deal with images and videos.

GaitSet[11], proposed by Hanqing Chao, Yiwei He, Junping Zhang and JianFeng Feng, in Dec 2018 is a deep learning based gait recognition system in which they regard gait of each subject as a set consisting of frames. Their system achieves an average rank-1 accuracy of 95.0% on CASIA-B dataset and an 87.1% accuracy on the OU-MVLP dataset making their results the new state-of-the-art at that time. Their model also shows

an accuracy of 87.2% and 70.4% on CASIA-B under bag-carrying and coat-wearing walking conditions.

Pattern Recognition[12], proposed by Yuqi Zhang, Yongzhen Huang, Liang Wang and Shiqi Yu, in April 2019 is a gait recognition system based on a joint Convolutional Neural Network CNN which achieves state-of-the-art performance of that time and their system also predicts soft biometrics such as gender and age from gait data as well with promising results.

Hundreds of gait recognition systems have been proposed in the last decade that naming all of them would require a complete book. According to my research, these two gait recognition systems provide the best results so far and can be considered state-of-the-art. Therefore, I will be comparing our proposed solution with these two solutions.

1.3 Gait Databases

Because of extensive research in this area of computer vision and biometrics, a lot of gait data has been collected and organized in the form of databases by various universities and companies. Many of these databases have been made public for research purposes. Since my model is based on videos, I will only be talking about video-based gait databases.

The most famous gait database is the CASIA[13] dataset published by Wang et al. in 2003. It consists of 3 datasets, CASIA-A, CASIA-B and CASIA-C. Out of these 3 datasets, CASIA-B is the most used which consists of 124 subjects, each recorded at 11 different angles and four walking variations e.g. normal walk, carrying a bag, winter clothing, summer clothing.

CMU[14] dataset was published in 2004 which consisted of 25 individuals walking on a treadmill in a 3D room. Gait of these individuals are recorded at 4 walking styles i.e. slow walk, inclined walk, fast walk and walk with a ball.

The Multi View Large Population dataset OU-MVLP[15] was released by Osaka university. This the largest gait dataset consisting of 10,307 subjects, both male and

female of different ages, recorded at 14 different angles by seven cameras. The dataset is already divided into two disjoint sets of testing and training of almost same size for research purposes.

1.4 Problem Definition

21st century is an era of technology; everything is being transferred to the computers or online from personal information of a person to confidential secrets of agencies to record of daily handlings by people or companies. Each individual is storing all of their data on digital systems because as humans we always look for ease of access. Due to this, security of these digital systems and places where data is stored has become one of the major concerns. Previously, people used traditional security measures such as pin, patterns, passwords etc. but as computers are becoming more and more advanced, these traditional measures are becoming easy to crack.

Problem is not only limited to the security of digital devices but also involve the security of physical assets and places. As mentioned earlier, the traditional security techniques such as passwords are easy to break through hence are not so reliable. Employing security guards everywhere also requires lot of human workforces. Verification and authentication of each individual manually becomes a time-consuming process and thus is not efficient.

The solution to this problem is biometric authentication systems. Today's biometric authentication systems i.e., facial recognition, fingerprint scanner, voice recognition etc. require human cooperation and are limited to small distances only. Moreover, the biometric factors these systems depend upon are prone to theft and duplicity therefore, can easily be bypassed. For instance, a person's fingerprints can easily be copied from anything that person touched and it has become a common practice nowadays to steal someone's fingerprints. Similarly, most of the facial recognition systems can be bypassed using an image of the person. In the same way, voice of a person can be created by using a phone call recording of that person.

So, the question arises, does a biometric factor exists which is not prone to all the afore mentioned drawbacks and hacks? Can a biometric authentication system be built using that biometric factor which is as promising as the rest of the biometric authentication systems? The answer is Yes! Gait Recognition System is a biometric authentication system which uses gait of individuals for verification and authentication.

1.5 Methodology

In this section, I talk about the methodological approach used, methods of data collection and data selection and finally the methods of analysis.

1.5.1 Methodological Approach

This project aimed at not only solving the practical problem mentioned in the problem definition section, but I also performed detailed research on how to improve the already existing theoretical gait recognition systems. The methodology used by me for the research is standard in the respective field and does not require any justification. Researchers are always experimenting to find new and better ways of doing things using latest technologies available. In my case, performing gait recognition system in a better way mean improving the final accuracy obtained using the proposed model. An increase of only 0.01% in the final accuracy makes the proposed model best in that respective field.

1.5.2 Methods of Data Collection/Selection

In case of gait recognition, the data either consists of video files of people walking in front of camera or frame images of those videos. For my initial research, I used existing publicly available database CASIA-B. The dataset can be downloaded online from various websites. CASIA-B dataset consists of 124 subjects, both male and female, whose gait data is captured from 11 different camera angles, from 0 to 180 with increment of 36 after each. The subjects are made to walk in three different variations, first carrying a bag,

second in winter clothing, and third normal walk. The folders are organized as follows /subject_number/walking_condition/angle.

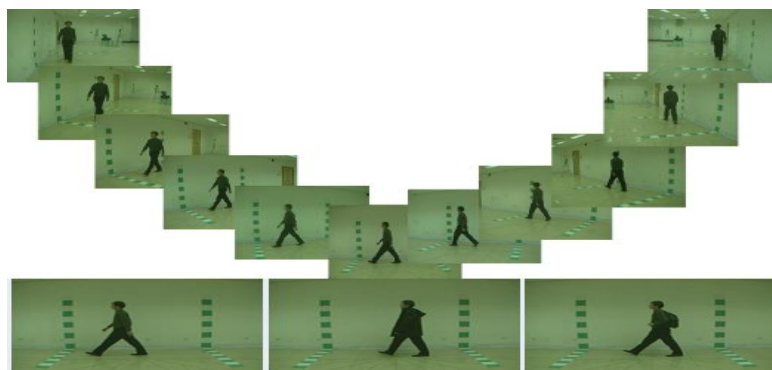


Figure 1.1- CASIA-B Dataset Example

Although the dataset also consists of actual video recordings, but I did not use them since video files are too large to handle and requires high-spec systems. I worked with the silhouettes extracted from video files.

After the initial testing, the next step is to collect my own data for real-world testing. For this purpose, I use a simple CCTV camera and subjects are made to walk in front of the camera various times. One restriction here is that the background should be clear, preferably white background, to produce better quality silhouettes and hence the final results. Majority of gait recognition systems uses CASIA-B dataset for testing, therefore I tried to replicate the CASIA-B dataset as much as possible. Gait of subjects are recorded from different angles and under different conditions and finally arranged in folders.

Due to an unfortunate situation in Ukraine, I was not able to collect the data using universities CCTV or use the GPU in our labs to run all the videos and extract the silhouettes and perform noise reduction, which was required for the frontend of GUI and GEI work as I had to return back to my country. To overcome that issue, I went for CCTV sample video and worked on it and performed the required functions and rest I did research work, this is provided my report.

1.5.3 Methods of Analysis

Before starting the testing and analysis of proposed system, gathered data is converted into the required form along with handling of outliers, checking for missed data and discarding of inappropriate data. Now, all the extracted silhouettes are not perfect and contains a lot of noise because of imperfect background. For better selection of data, I set a threshold value of white pixels. Since silhouette images consists of white and black pixels, with white pixels representing the human body and all the remaining portion of image as black pixels. If the total number of white pixels in a silhouette image is less than -enter value-, then that image is discarded and not used. Similarly, if the total number images for one complete walk of an individual are less than 20, then that data is discarded and the subject is made to walk again.

CHAPTER 2. DESIGN OF SYSTEM OF GAIT RECOGNITION

In this section, I talk about the architecture of my proposed solution and in-depth design of all the subsystems and subcomponents it is broken down into. The reasons for breaking down the system into respective subcomponents are also clearly stated. To get a general idea about the system, the architecture and subsystem architecture are shown in section 2.2.

2.1 Mathematical Fundamentals of System Design

Just like any other gait recognition system, my system also has four main phases involved. These are Data Acquisition, Preprocessing, Feature Extraction and Classification. Each phase involves a set of different techniques and algorithms each of which is accomplished using a different tool or software. Data acquisition is used to collect data which is then transformed into model specific information in Preprocessing phase. Important characteristics are then extracted from the data for recognition in Feature extraction phase and finally identification is performed in Classification phase.

2.1.1 Data Acquisition

The first step in every biometric authentication system, in fact in every system, is the collection of data. No system can work without the data. Now, there are various ways through which gait data can be collected. Naming a few, through video cameras, through pressure sensors installed in the ground, through accelerometers worn by the subjects, through 3D depth sensors to get a 3D model of the subject.



Figure 2.1 - CCTV Camera

The method which I chose for collection of data is through CCTV camera since the main application of my system is surveillance which is done through CCTV cameras all over the world. There is no special algorithm behind the acquisition of video data using cameras. A person simply walks in front of the camera and the video is recorded and saved on the system. However, this will produce gigabytes of data and data storage will become a problem. To overcome this, I used a simple Pedestrian Detection algorithm in combination with contours to detect persons walking. Pedestrian detection algorithm continuously run at the backend, whenever a person appears in front of the camera, the algorithm detects it and signals the system to start saving the video frames. Contour is simply a rectangular box along the body of the person specifying the subject as shown in the figure below.

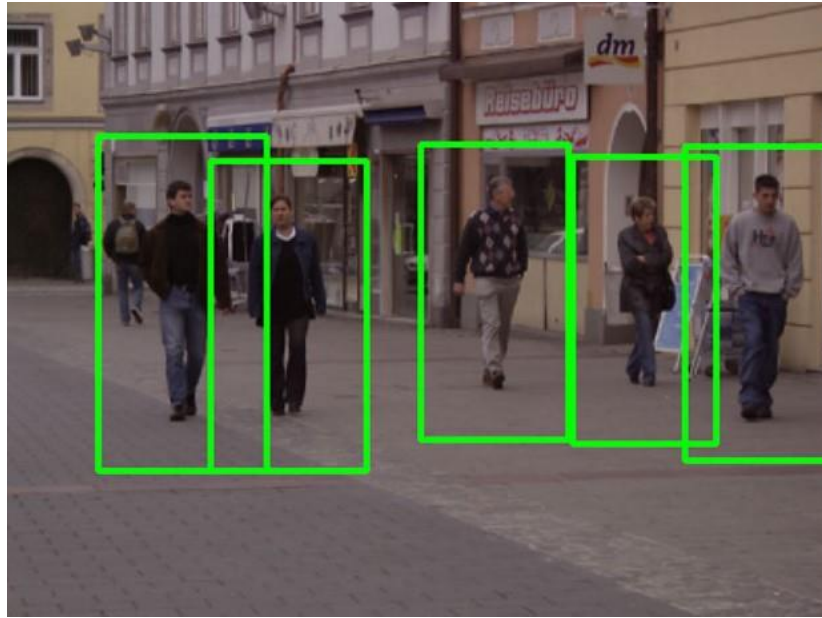


Figure 2.2 - Pedestrian Detection

Each subject is made to walk in front of the camera five or six times for proper data acquisition. To improve the results, gait of a person can be recorded at different angles to get various viewpoints. Results can be greatly improved if the background is completely white, the reason for which is associated with silhouettes discussed in next section.

2.1.2 Pre-Processing

Data collected in the first step is in raw form i.e., video files. Next step is to convert that data into a form specific for our system. For this purpose, I first convert the saved video files into frames since it is easy to operate on images as compared to video files. A 30 seconds walk in front of the camera, from start to end, approximately produces 60-90 frames which are enough to preserve the gait data.

Each frame is converted from RGB to grayscale to reduce the number of pixels. Since we are not concerned with details of what the person is wearing, a technique called Silhouette Extraction is used. Silhouette is simply an illustrated outline of the body with foreground, i.e., body, in white pixels and background in black. It is because of this reason that each frame is converted into grayscale and also the reason why white background is used while data acquisition. After silhouette extraction, I get these images



Example of silhouette extraction

Figure 2.3 - Silhouette Extraction

Now, because of the feature extraction technique used (explained in next section), another problem arises. That is the position of silhouette in each frame. Since the person walks from one to the other comprising one gait cycle, silhouette in the beginning of frames is in the start of image, then in the middle of image and finally transitioning to the end of image as a person completes his/her walk-in front of camera as shown in the image below



Figure 2.4 - Silhouette Extracted Images before cropping and resizing

This is overcome by again using Pedestrian Detection Algorithm to detect the silhouette, crop that part of image, resize to a specific size so that all images are of same size and finally saved in a different folder.

Since the background is not always perfect and also the quality of image from a simple CCTV camera is not high, there is a lot of noise in these extracted silhouette frames. I remove this noise using a combination of different morphological operators. After all of the pre-processing, the final image obtained is

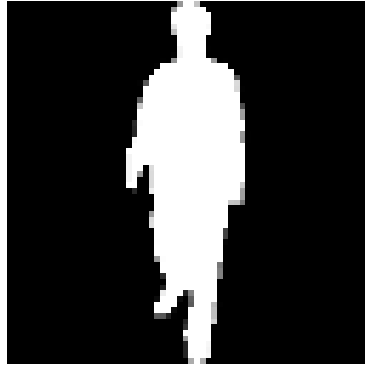


Figure 2.5 - Output after complete Pretreatment

2.1.3 Feature Extraction

As the name suggests, the next step is to extract features from pre-processed frames in order to recognize individuals. Different feature extraction methods have been proposed by researchers in the past, each having its own advantages and disadvantages. There are two approaches for extracting gait features, model-based and model-free. Model-based approach makes a model of the human body and then extracts features from the proposed model. Model-free approaches focuses more on the silhouettes or the whole motion of human bodies instead of human body models.

I use a model-free approach Gait Energy Image (GEI). In this technique, frames of one gait cycle are superimposed on each other. This produces a single image which nothing but the accumulative energy image of all the images of that person for that specific walk. In simple words, 60-90 frames of each individual walk at a specific angle are converted into 1 image which can be regarded as unique signature of that person's gait.



Figure 2.6 - Frames superimposed on each other to calculate GEI

Given a sequence of silhouette frames $S(x, y, t)$, where x and y are coordinates in the frame and t is the frame index, GEI is computed as follows:

$$GEI(x, y) = \frac{1}{N} \sum_t^N S(x, y, t)$$

where N is the total number of frames.

Gait Energy Image retains the temporal and spatial features of an individual's gait. A pixel having high-intensity ,i.e. more white, indicates frequent activity at that point.

2.1.4 Classification

The final step is the identification of subjects based on their Gait Energy Images GEIs. There are different methods for classification as well. I use deep learning in my system as a research part as it requires a GPU to run all images, I did this part as research and how it could be implied in my system. There are various deep learning and machine learning models used for classification of image, each having their own state-of-the-art accuracies.

I did the research on the famous ResNet34 model which Convolutional Neural Network (CNN) consisting of 34 layers. It is the backbone of many computers vision

tasks and has achieved record-breaking results in image classification task. Since my final GEI after feature extraction is an image having pixel values, ResNet34 can be easily used for identification.

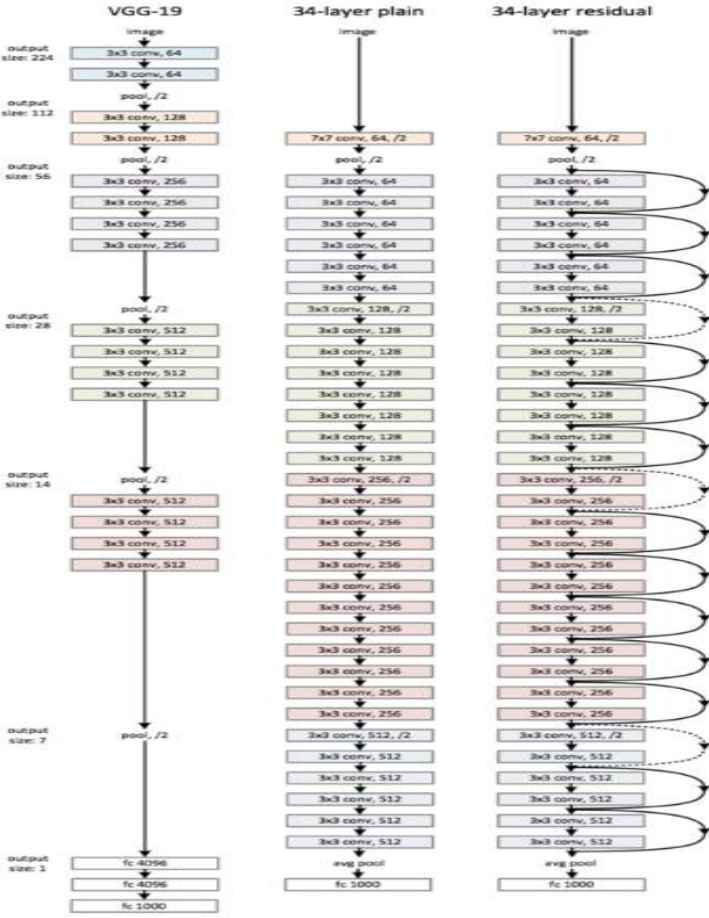


Figure 2.7 - ResNet34 Architecture

2.2 System Architecture

2.2.1 Architecture Design Approach

The whole architecture of the proposed system was designed keeping in view the constraints and the scope of the project hence leading to research part only. In some places, to avoid reinventing the wheel, already proposed algorithms are used for efficiency and reliability while in other places, new algorithms are designed specific for the system and data.

2.2.2 Architecture Design

The following figure depicts the overall structure of the system along with subsystems.

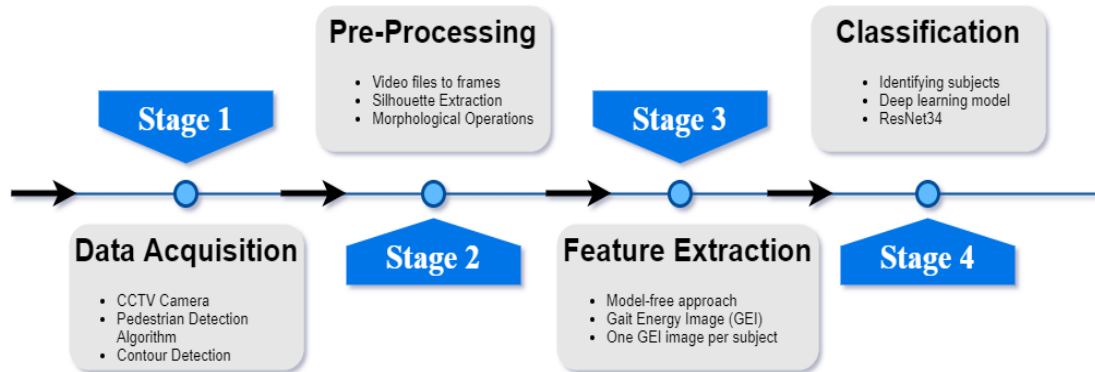


Figure 2.8 - Division into subsystems

2.1.3 Subsystem Architecture

Each of the 4 subsystems are further divided into modules, the in-depth details of which are presented in the next section. This section gives an overview of what these modules are and their architecture design.

First subsystem Data Acquisition is divided into two main modules, namely Camera and Pedestrian Detection for capturing data and identifying subjects in the data respectively. Pedestrian Detection has an optional submodule, Non Max Suppression (NMS), for better performance and results.

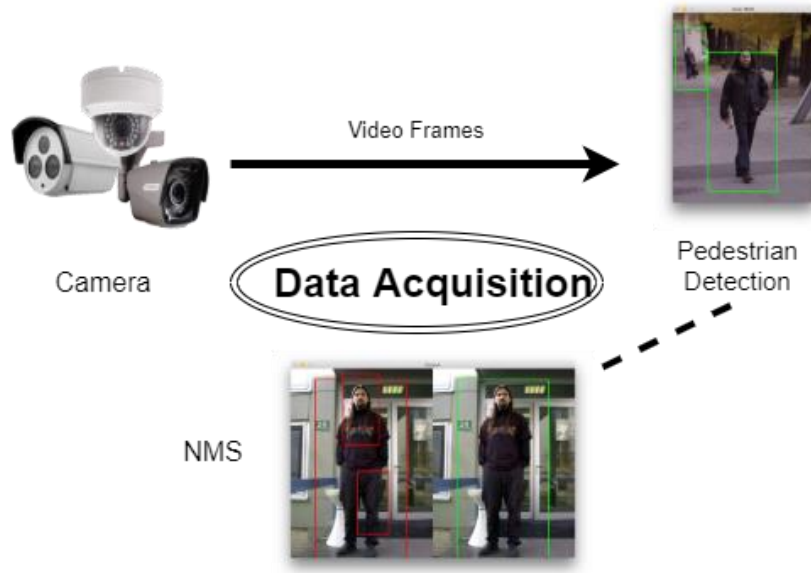


Figure 2.9 - Data Acquisition modules

The second subsystem Preprocessing is a complete algorithm in itself consisting of two main modules, Silhouette Extraction and Resizing. The in-depth detail of both of these modules is present in the next section.

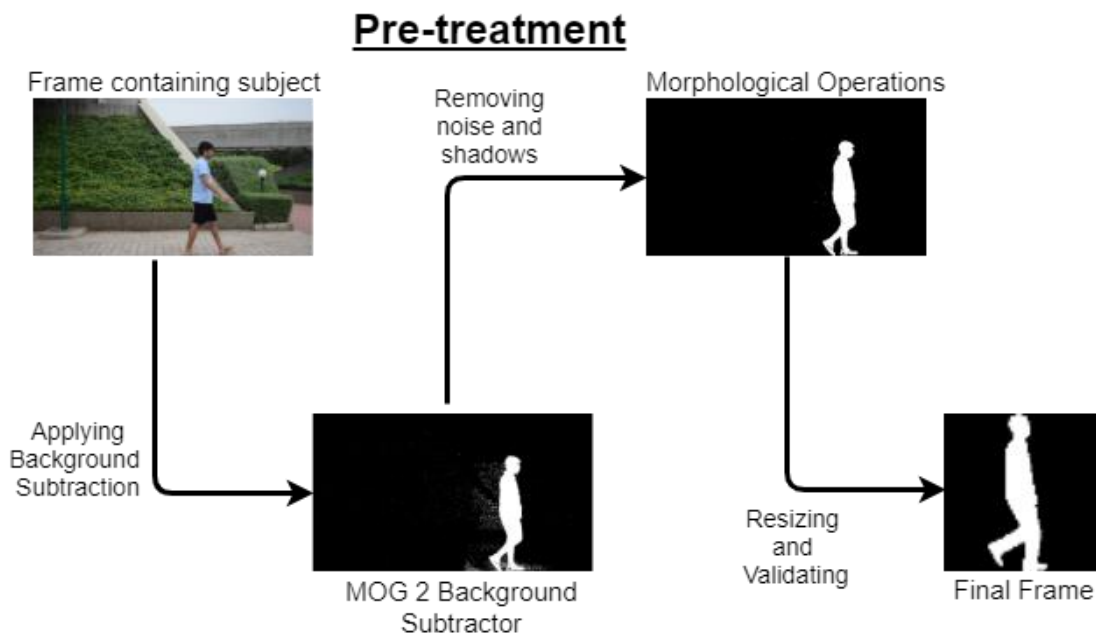


Figure 2.10 - Pretreatment modules

2.3 Detailed System Design

This section talks about the details of each module of the subsystems and also explains the further classes and functions these modules depend upon as well as the constraints and restrictions on each subsystem and their modules.

2.3.1 CCTV Camera

Classification

CCTV Camera used is the main module of first subsystem Data Acquisition. It is solely a hardware component.

Definition

The only purpose of CCTV camera is to record data. Although it is recording 24/7, but it only starts saving the video when it detects a person in the video. Detection of the individual is performed by another module named Pedestrian Detection.

Constraints

There are certain limitations in the working of CCTV camera in our model. The main issue with the storage, since video files are usually of large sizes. Storing of 24 hours surveillance videos is not feasible and storage friendly. This puts a constraint on camera to only start saving the video when directed or under specific conditions.

Composition

This module has no further subcomponents.

Uses/Interactions

CCTV camera is connected to the main system, on which the complete database of gait data of individuals is present, either directly or through server in case of remote surveillance.

Resources

The only resource needed by the CCTV camera is continuous source of power either through batteries or electricity.

2.3.2 Pedestrian Detection

Classification

This is the module [\[16\]](#) of first subsystem Data Acquisition.

Definition

As the name suggests, this module is used to detect the presence of humans in the video recorded by the camera.

Responsibilities

This module plays a major role in first subsystem and is also used to overcome the constraint of previous module CCTV camera. Whenever an individual walks in front of the camera, it immediately detects that a person is walking and sends signal to the first module to start saving the video now.

Constraints

This module requires a GPU for quickly loading the pedestrian detection pre-trained model and initializing the detector. Also the CPU must be fast since this module is run once every 27 milliseconds. If a GPU is not present, then simple background subtraction can achieve the same functionality however, background subtraction puts another constraint on the background area where the individual is walking and requires a complete white background for accurate detection.

Composition

This module does not contain a subcomponent. However, optionally, Non Maxima Suppression (NMS) algorithm can be applied for better performance.

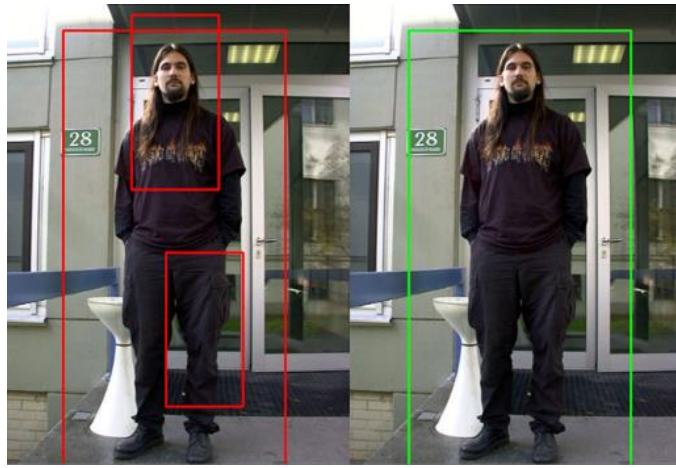


Figure 3.11 - Before and After applying NMS

Uses/Interactions

The input of this module is directly connected to the output of CCTV camera, without video data there is nothing to detect. The output of this module is the final output of first subsystem Data Acquisition, that is video files in which individual is walking in front of the camera. If the quality of video provided by the camera is not good, then this module will not work perfectly.

Resources

This module requires high-speed processors and an optional GPU for faster processing. It works for any version of Python i.e. python2.7 or python3.x. Python library opencv is required by this module. It works for opencv version 2.4.X and 3. An optional Non Max Suppression library, present in imutils.object_detection, is required for better performance.

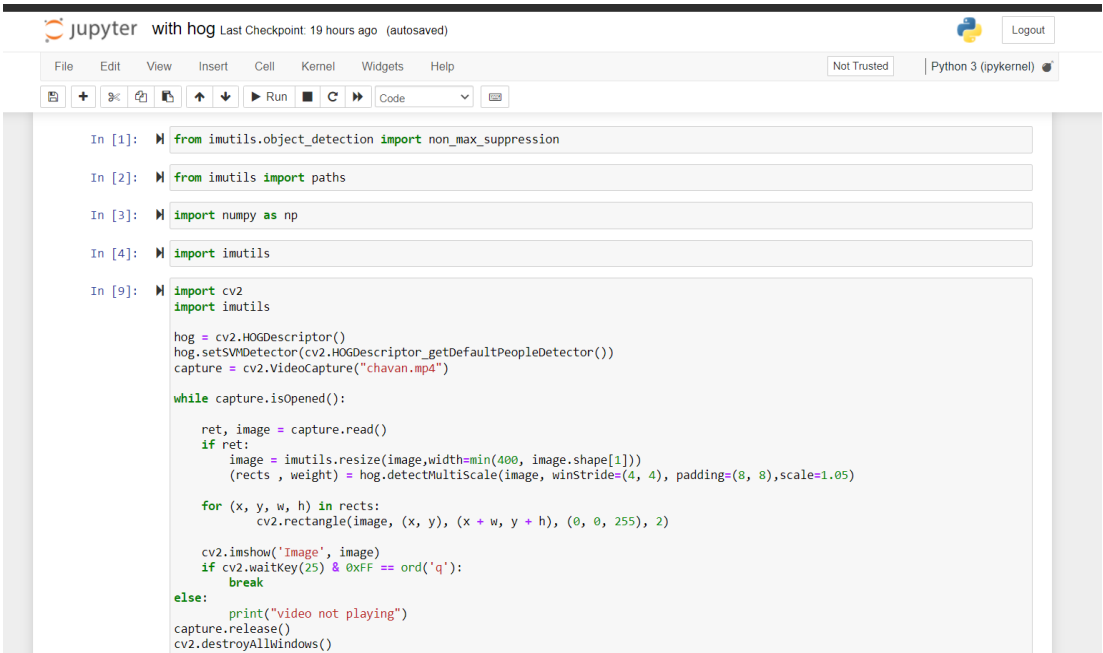
Processing

First of all, it takes video being recorded by the camera as input in the form of frames. Then HOG descriptor (Histogram of Oriented Gradients) is initialized. After that, SVM detector is set to an already pre-trained pedestrian detection dataset present in the cv2 library. This initializes the pedestrian detector which is then applied on each frame. Whenever an individual walks in front of the camera, it detects it and draws a bounding box around the individual. Optionally, the non-maxima suppression algorithm can be

applied to the bounding box to maintain overlapping of boxes. Finally, the resulting frames are displayed. The detectors are initialized only once at the start when the system powers up.

Interface

```
1 #Importing Libraries
2 from imutils.object_detection import non_max_suppression
3 import numpy as np
4 import cv2
5
6 #Capture video
7 capture = cv2.VideoCapture("Enter name of video file")
8
9 #Initializing the HOG person detector
10 hog = cv2.HOGDescriptor()
11 hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
12
13 success, frame = capture.read() #Read the first frame
14 #Looping over all the frames
15 if success:
16     (rects, weights) = hog.detectMultiScale(frame, winStride=(4,4),
17                                             padding=(8,8), scale=1.05)
18
19     #Drawing the bounding box
20     for(x,y,w,h) in rects:
21         cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 2)
22
23     success, frame = capture.read() #Get the next frame
```



```
In [1]: from imutils.object_detection import non_max_suppression
In [2]: from imutils import paths
In [3]: import numpy as np
In [4]: import imutils
In [9]: import cv2
import imutils

hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
capture = cv2.VideoCapture("chavan.mp4")

while capture.isOpened():
    ret, image = capture.read()
    if ret:
        image = imutils.resize(image,width=min(400, image.shape[1]))
        (rects , weight) = hog.detectMultiScale(image, winstride=(4, 4), padding=(8, 8),scale=1.05)

        for (x, y, w, h) in rects:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

        cv2.imshow('Image', image)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    else:
        print("video not playing")
    capture.release()
cv2.destroyAllWindows()
```

Figure 2.12 - Code for video to image transformation

First necessary packages are imported. Then cv2 method VideoCapture(0) is used to start recording video. Instead of video file name, if other cameras are attached to the system, the argument can be changed accordingly to 0, 1 or 2. Next lines initializes the detectors accordingly. Then I start looping over the frames. cv2.read () method is used to read frames from the video. Actual detection is performed by hog.detectMultiScale. The first argument is the frame, second argument signifies the step size of kernel window which is 4x4 pixels in my case. This means that kernel window will move 4 pixels in both x and y direction. Last argument 'scale' indicates the layers in the image pyramid, if it is too small it results in false-positives and if it is too large, it might not detect pedestrian at all. detectMultiScale returns x and y coordinates of the pedestrian in the frame which are then used by cv2.rectangle to create a bounding box around the pedestrian.

If NMS is to be used, just add the following lines of code

```
19     #Applying NMS
20     rects = np.array([[x,y,x+w,y+h] for (x,y,w,h) in rects])
21     pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)
22
23     #Drawing the bounding box
24     for(x,y,w,h) in picks:
25         cv2.rectangle(frame, (x,y), (w, h), (0,0,255), 2)
26
```

Figure 2.13 - Code for applying contour

Image



Figure 2.14 - Image after contour added (red box)

Detailed Subsystem Design is given on Fig 2.15

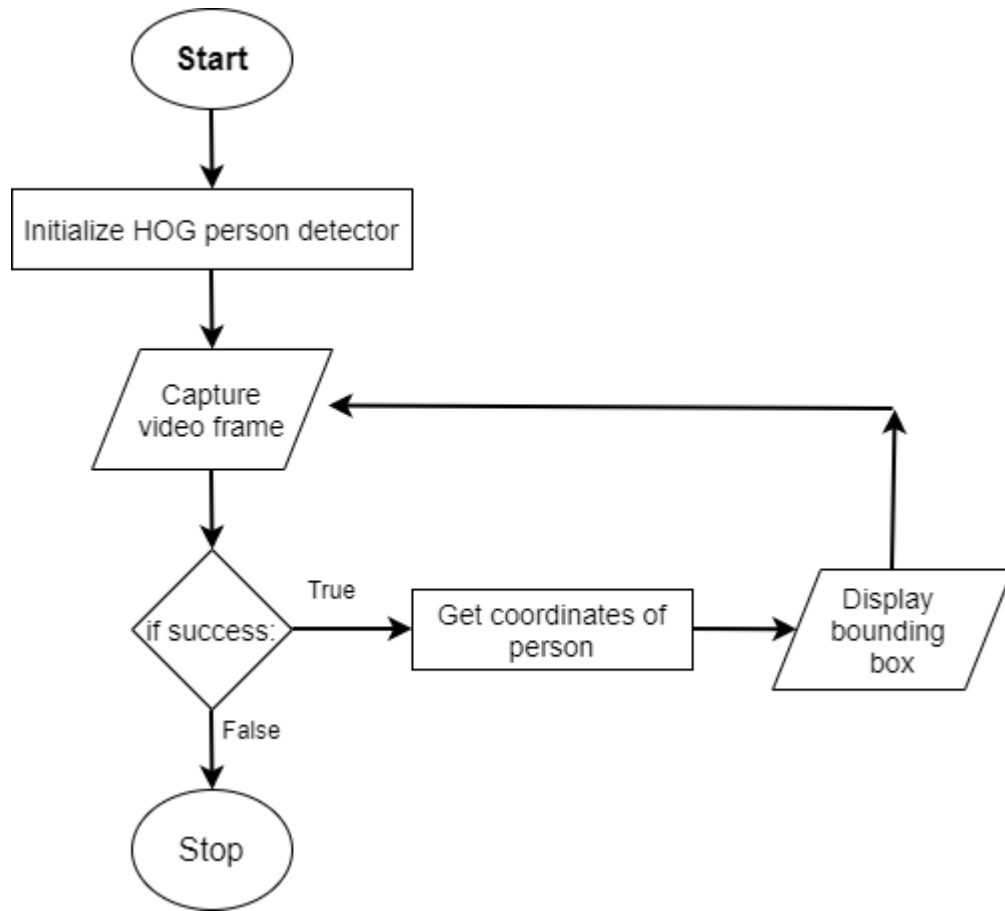


Figure 2.15 - Background Subtraction design

CHAPTER 3 IMPLEMENTATION AND TESTING

3.1 Pre-Processing Algorithm Implementation

Classification

It is a complete subsystem itself.

Definition

It is used for processing of video frames, captured by camera, to transform them into frames specific for our model and also to improve the final accuracy.

Responsibilities

This subsystem plays a vital role in the whole system. Although, the system works normally without pre-processing, but it may produce lower quality results and false-positives. To overcome these and to improve the efficiency of the system, this algorithm is used. The primary responsibilities of this algorithm is to extract silhouettes from video frames and then crop and resize those extracted silhouettes so that each silhouette is in the center of every image.

Constraints

If the data to be processed is very large, then there must be a GPU present since we are dealing with thousands of images. In the absence of GPU, the algorithm still runs but it takes a lot of time reducing the efficiency of the whole system.

Composition

The algorithm consists of three main modules; first is Silhouette Extraction[\[17\]](#) which is used to extract silhouettes from the video frames, second is Valid Frames which checks whether the extracted silhouette images contain enough information, and the third is Resizing which crops every valid silhouette image and resize it so that all images contains silhouette at the same coordinates and are of same size.

Uses/Interactions

The input of the algorithm is the output of first subsystem data acquisition that is the video frames. This algorithm runs in parallel with the first subsystem, its performing

silhouette extraction every second, but only starts saving the frames when the first subsystems signal the presence of an individual in the frame. After the individual has completed his/her walk in front of the camera, the final output of the algorithm is then fed to the next subsystem Feature Extraction.

Resources

This module requires high-speed processors and an optional GPU for faster processing. It works for any version of Python i.e. python2.7 or python3.x. Python library opencv is required by this module. It works for opencv version 3.X or higher.

Processing

First module, Silhouette Extraction, uses an algorithm Background Subtractor MOG2 which is based on two papers published in 2004[\[18\]](#) and 2006[\[19\]](#) respectively. The extracted silhouette frames consist of noise which is removed using morphological operation Opening[\[20\]](#) and applying a threshold for removing silhouettes of shadows in the images.

Then the Valid Frames module checks whether the sum of white pixels in silhouette frame is greater than 10,000, if it is not then this indicates that this frame is not valid for identification and is discarded from the data.

The Resize module then takes the valid silhouette frames and crop them so that the silhouette is in the center of each frame and also resize every frame on the basis of height and width of person in the frame so that final image does not appear distorted.

Finally, the algorithm checks whether the total number of valid silhouette frames in one gait cycle of individual are greater than 5 or not. If they are less than 5, then data is too less for identification and it prompts the system to re-enter the data for that individual.

Interface/Exports



```

In [1]: import numpy as np
import cv2

capture = cv2.VideoCapture("chavan.mp4")
fgbg = cv2.createBackgroundSubtractorMOG2()

while(1):
    ret, frame = capture.read()

    fgmask = fgbg.apply(frame)

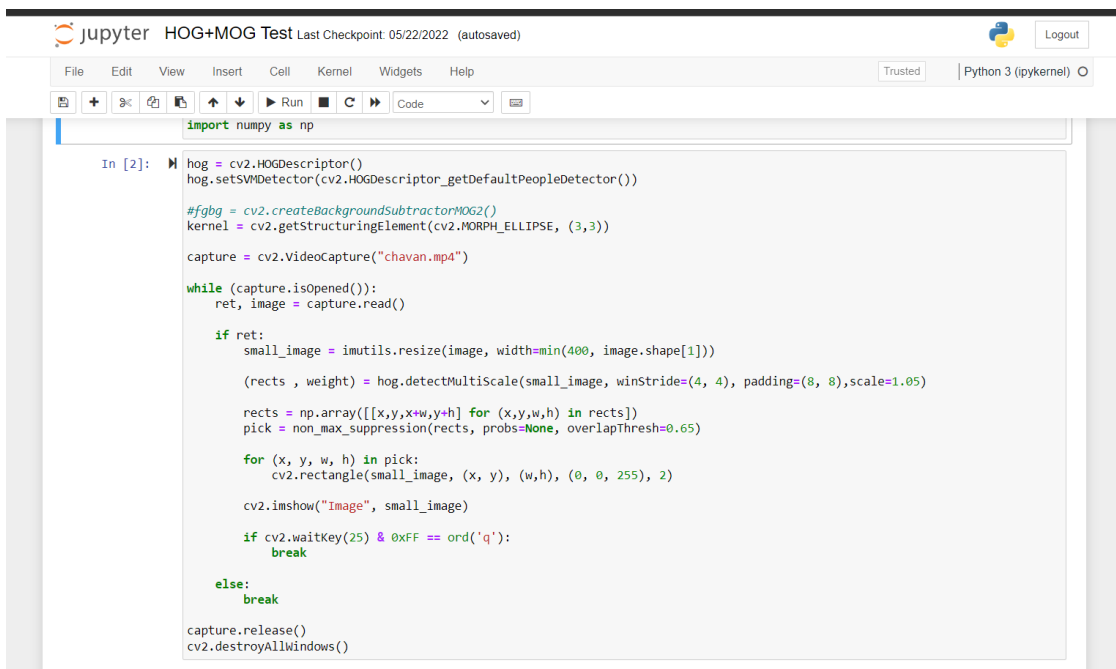
    cv2.imshow("fgmask", fgmask)
    cv2.imshow("frame", frame)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

capture.release()
cv2.destroyAllWindows()

```

Figure 3.1 - Code for applying background subtractions



```

import numpy as np

In [2]: hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

#fgbg = cv2.createBackgroundSubtractorMOG2()
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))

capture = cv2.VideoCapture("chavan.mp4")

while (capture.isOpened()):
    ret, image = capture.read()

    if ret:
        small_image = imutils.resize(image, width=min(400, image.shape[1]))

        (rects , weight) = hog.detectMultiScale(small_image, winStride=(4, 4), padding=(8, 8),scale=1.05)

        rects = np.array([[x,y,x+w,y+h] for (x,y,w,h) in rects])
        pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)

        for (x, y, w, h) in pick:
            cv2.rectangle(small_image, (x, y), (w,h), (0, 0, 255), 2)

        cv2.imshow("Image", small_image)

        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

    else:
        break

capture.release()
cv2.destroyAllWindows()

```

Figure 3.2 - Code for applying grayscale and resizing the frame

```

21 self.backgroundMask = cv2.createBackgroundSubtractorMOG2()
22 self.kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))

```

For using Background Subtraction MOG2 algorithm, cv2 internal function is used which contains algorithm specifications. In the next line, kernel is initialized using cv2.getStructuringElement. First argument specifies the shape of kernel and second argument specifies the size of kernel, which in my cases are ellipse and 3x3 respectively.

```
198     thresh = self.backgroundMask.apply(frame)
199     thresh = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, self.kernel)
200     thresh = cv2.threshold(thresh, 200, 255, cv2.THRESH_BINARY)[1]
201
```

MOG2 algorithm is applied to each frame using `.apply(frame)` method and the morphological operation Opening is applied to the resulting frame using `cv2.morphologyEx` using the frame, morphological operation and kernel as arguments. This to remove unwanted noise in the frame. Finally, shadows are removed using `cv2` method `.threshold()`. The arguments specifies that all pixels with values greater than 200 will be turned to 255 or white.



Figure 3.3 - Image after MOG2 and turned into grayscale

Now I add noise removal code to make the image more clear

```
jupyter Noise-Removal Last Checkpoint: 8 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
In [4]: import numpy as np
import cv2

capture = cv2.VideoCapture("chavan.mp4")
fgbg = cv2.createBackgroundSubtractorMOG2()
success, frame = capture.read()
count = 0
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))

while success:
    fgmask = fgbg.apply(frame)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
    fgmask = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)[1]

    cv2.imshow('fgmask', fgmask)

    cv2.imwrite("frame%d.jpg" % count, fgmask)
    success, frame = capture.read()
    count += 1

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

capture.release()
cv2.destroyAllWindows()
```

Figure 3.4 - Code for applying Noise removal

When we need elliptical/circular shaped kernels, for this purpose, OpenCV has a function, **cv2.getStructuringElement()**.

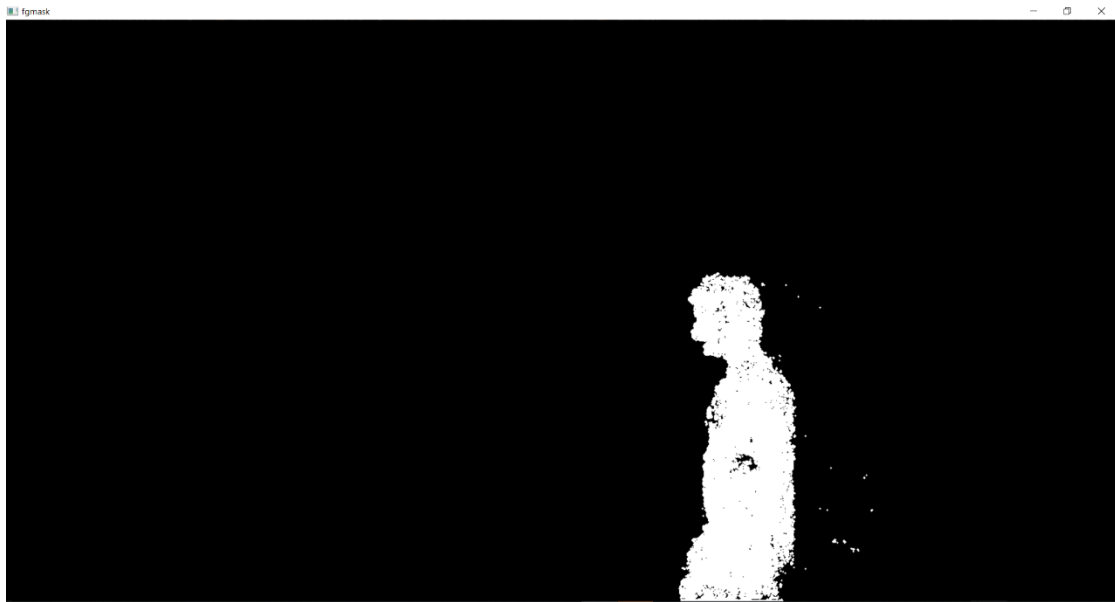


Figure 3.5 - Image getting clear after noise removed

The code for pre-treatment is given at the end of the report

```

# Warn if the sequence contains less than 5 frames
print(count_frame)
if count_frame < 5:
    message = 'seq:%s, less than 5 valid data.' % (
        '-'.join(seq_info))
    warn(message)
    log_print(pid, WARNING, message)

```

This piece of code is used to warn if the gait cycle of an individual contains less frames, so that gait can be recorded again.

```

# A silhouette contains too little white pixels
# might be not valid for identification.
if img.sum() <= 10000:
    message = 'seq:%s, frame:%s, no data, %d.' % (
        '-'.join(seq_info), frame_name, img.sum())
    warn(message)
    log_print(pid, WARNING, message)
    return None

```

```

def log_print(pid, comment, logs):
    str_log = log2str(pid, comment, logs)
    if pid != 0:
        print("\n" + str_log)
    if comment in [WARNING, FAIL]:
        with open(LOG_PATH, 'a') as log_f:
            log_f.write(str_log)
    if comment in [START, FINISH]:
        if pid % 500 != 0:
            return
    print(str_log, end='')

```

This method is used to record everything happening during the pretreatment process in a log file. An example snapshot of log file is


```

Pretreatment Start.
Input path: C:/Users/Uqqasha/Desktop/FYP/CASIA-B
Output path: C:/Users/Uqqasha/Desktop/FYP/Pretreatment Output
Log file: C:/Users/Uqqasha/Desktop/FYP/pretreatment.log
Worker num: 1
# JOB 0 : --START-- 001-bg-01-000
89
# JOB 0 : --FINISH-- Contain 89 valid frames. Saved to C:/Users/Uqqasha/Desktop/FYP/Pretreatment Output\001\bg-01\000.

# JOB 1 : --START-- 001-bg-01-018
91
# JOB 1 : --FINISH-- Contain 91 valid frames. Saved to C:/Users/Uqqasha/Desktop/FYP/Pretreatment Output\001\bg-01\018.

# JOB 2 : --START-- 001-bg-01-036

```

Detailed Subsystem Design

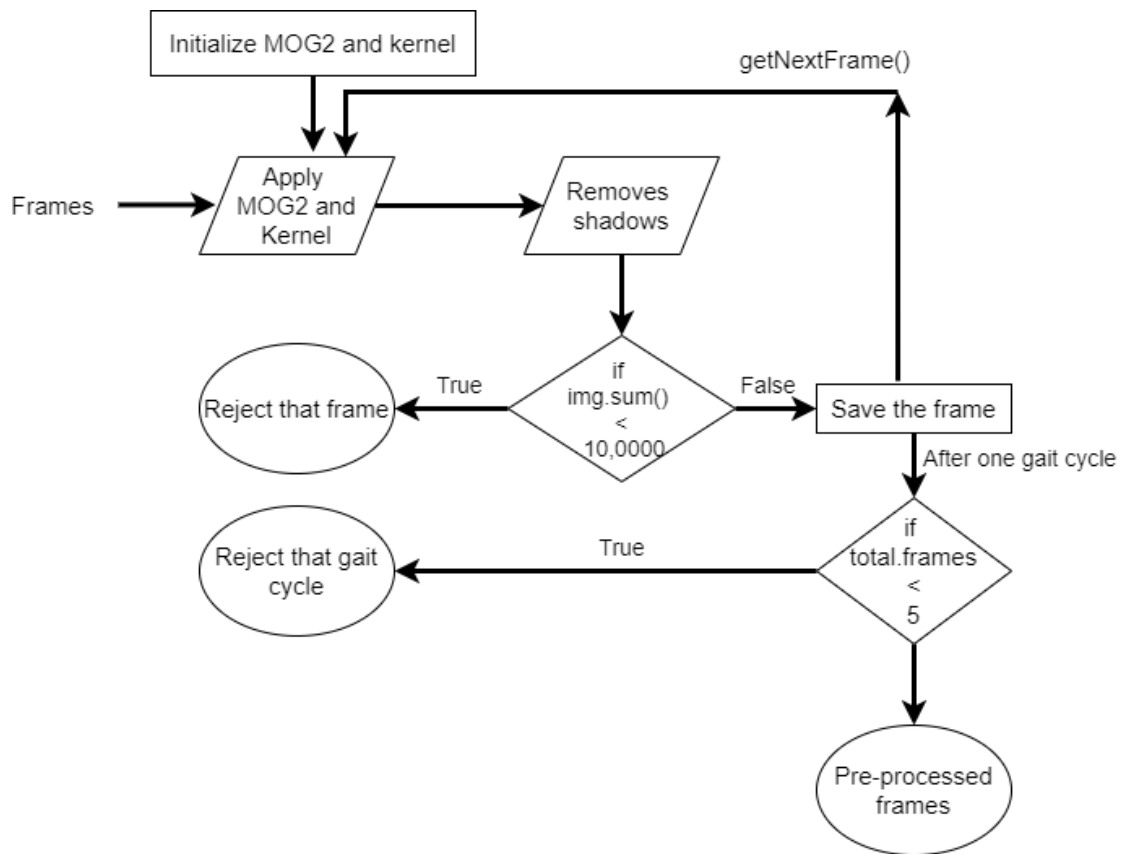


Figure 3.6 - Pretreatment algorithm design

3.2 Gait Energy Image (GEI)

Classification

This part is under research part and this is the algorithm used in subsystem Feature Extraction.

Definition

Gait Energy Image is used as a feature extraction algorithm which extracts unique features from silhouettes of an individual that can be used for the identification of an individual.

Responsibilities

Since it is the only algorithm used in the third subsystem, the system cannot perform detection without it. The primary responsibility of this algorithm is to take all the silhouette extracted frames of an individual for a certain sequence and angle, and calculate a single silhouette frame which can be used as a unique identifier for that individual and that final frame is called GEI for that individual.

Constraints

To obtain best quality GEIs, it is strongly recommended that the silhouette extracted frames be cropped in a such a way that silhouette is in middle of every frame. If the silhouette is at different positions in each frame, as shown in the image below, then the GEIs calculated will strongly affect the final identification.



Figure 3.7 - Silhouette Extracted Images before cropping and resizing

Such frames will produce the following GEI,



Figure 3.8 - Noisy GEI

Composition

This algorithm does not consist any further subcomponent or module.

Uses/Interactions

The input to the algorithm is preprocessed silhouette extracted valid frames which are the output of previous subsystem Pre-processing. The algorithm is repeated for each angle and each walking variation for an individual. For example, if there are 10 different walking variations and 11 different angles for each variation, then this algorithm will return $10 \times 11 = 110$ GEIs for that individual. These GEIs are then used as an input for final identification.

Resources

This module requires high-speed processors and an optional GPU for faster processing for that reason I did this as a research part. It works for any version of Python i.e. python2.7 or python3.x. Python library opencv is required by this module. It works for opencv version 3.X or higher. Other python libraries required includes OS, Numpy and Matplotlib.

Processing

What GEI algorithm does is it takes all the silhouette extracted frames and superimpose them on each other to get a single GEI frame. This final GEI frame can be thought of as an average of all the frames.

Given a sequence of silhouette frames $S(x, y, t)$, where x and y are coordinates in the frame and t is the frame index, GEI is computed as follows: -

$$GEI(x, y) = \frac{1}{N} \sum_t^N S(x, y, t)$$

Where N is the total number of frames. The resulting GEI obtained looks like this,



Figure 3.9 - Gait Energy Image (GEI)

3.3 Graphical User Interface (GUI)

Classification

This is the frontend of our final product for which I wrote the code which is given at the end of the report but it will require a GPU to run it as the video files are large and it could lead to slow outcomes and overheating of laptops.

Definition

The GUI is the frontend interface with which the users interact.

Responsibilities

GUI shows a visual representation of what is going on in the system to the user. It displays the video which is being captured by the CCTV Camera on the screen as well as

the result of background subtraction after noise and shadow removal. It allows the user to register a new person into the system, save the gait data of an individual in the system, shows the number of persons registered into the system and finally displays the name of the person when he/she walks in front of the camera.

Constraints

First of all, the name of the person should be registered into the system before saving their gait data otherwise the system will display an error that person is not registered into the system. Similarly, to recognize an individual, his/her gait data must be present in the system so that system can recognize him/her correctly and display their name. In terms of processing power, storage and timing, this component has no constraints.

Composition

GUI consists of 5 buttons with which user can interact. These are “Register” button which is used to register a new person’s name in the system. “Recognize” button which is used to identify the person walking in front of the camera. “Save” button which is used to store gait data of individual walking in front of the camera. “Update” button which is used to update the background subtraction algorithm and an “Exit” button used to shutdown the system.

Uses/Interactions

Since it is the graphical interface, it is used by almost all the other subsystems and modules. The result of first subsystem Data Acquisition is displayed on the screen in special window along with another window showing the result of module Background Subtraction. Similarly, after final identification of person by the subsystem Classification, the name of individual is displayed on the window which shows the video of person walking in front of the camera.

Resources

The GUI is built using python library PyQt5[\[21\]](#). It requires python version 3.6 or higher. However, a similar GUI made using PyQt4 and python 2.7 is also available and can be used, but since PyQt5 is the latest version so it is recommended.

Processing

First of all, an XML file is created with specific syntax which acts as a blueprint for the graphical user interface. All the components of GUI, such as Main Window, Central Widget, Menu Bar, Textboxes, and Buttons are defined in this file. A snapshot of this XML file is shown. After that python code is written using PyQt5, which loads this XML file and sets up the frontend. Although, the specifications of all the components can be made through python code as well, and some examples of doing that are also provided in the .py file, but it is recommended to use the XML file to keep the code minimalistic. All the functions which executes when a certain button on the screen is pressed are defined in the .py file and a special play() method which is connected to a timer. Timeout value of the timer can be set. If timeout value is 27, this means that play method will be executed after every 27 milliseconds.

Interface/Exports

The final interface of the GUI looks like this,

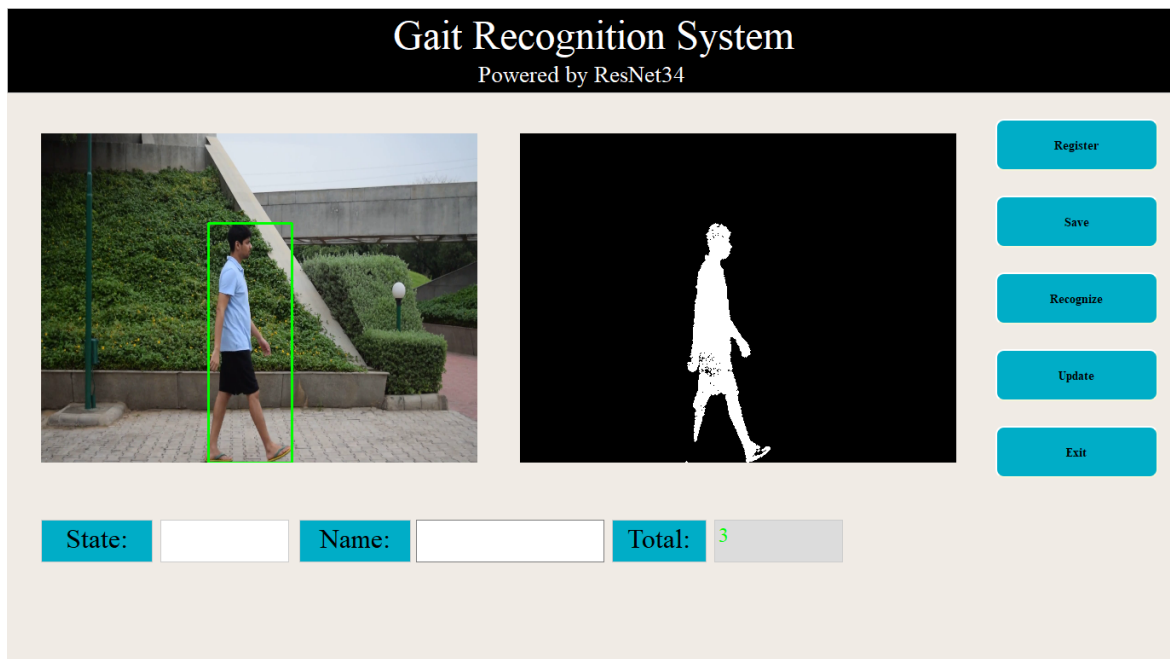


Figure 3.10- GUI

The video on the left side is the direct output of camera while the video on the right side is after performing background subtraction and noise removal.

Detailed Subsystem Design

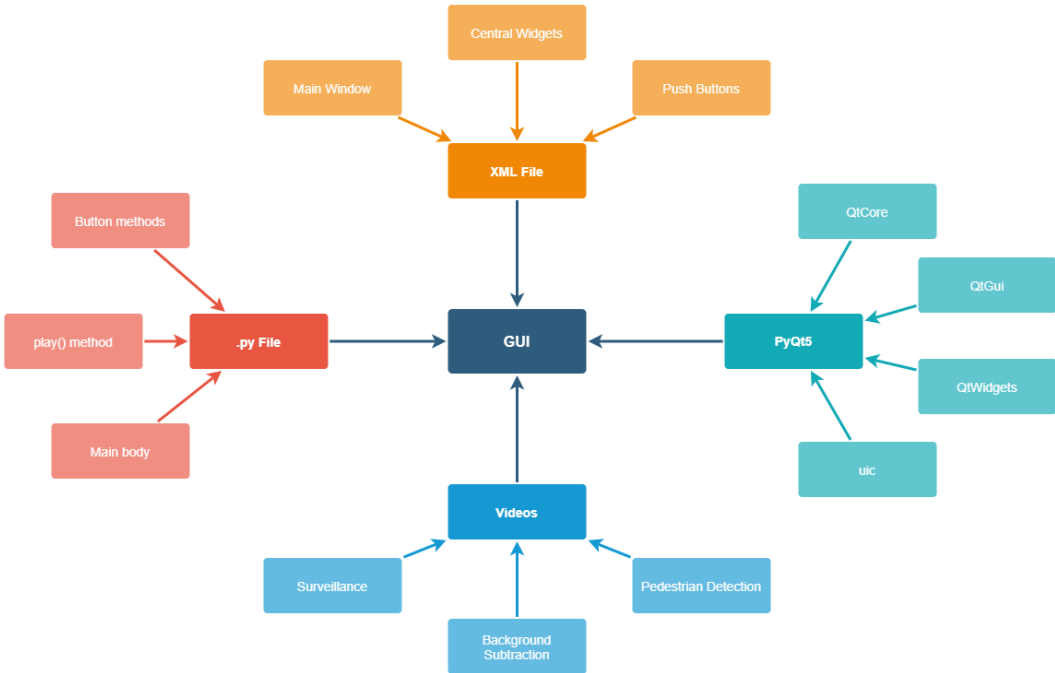


Figure 3.11 - GUI Map, description on how to create the frontend of the GUI

3.4 Class Diagram

In this section I researched about the implementation of the system discussed earlier. The heart of our system is ResNet (Residual neural network) which is a CNN (Convolution Neural Network). I implemented ResNet using Fastai [22] which is a Python library and one of the world’s most popular AI framework. As mentioned earlier I used Casia-B dataset to test and implement ResNet for Gait recognition.

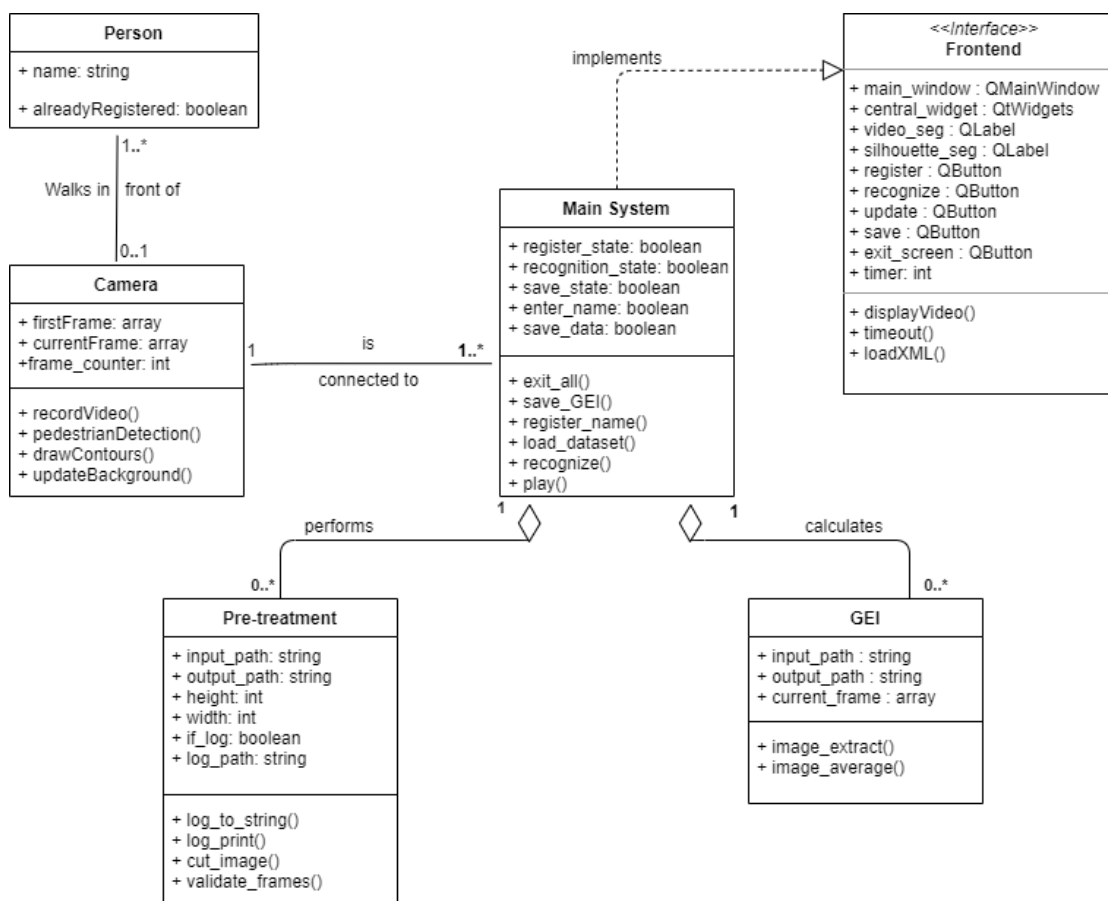


Figure 3.12 - Class Diagram, describing how many classes were used in the codes for pre-treatment, GUI, GEI and how the camera registers the person.

3.5 Training And Testing Resnet

Before the implementation of Resnet in my system I tested it and compared its results with the state-of-the-art methods for gait recognition for this again a GPU is required as CASIA-B has a large amount of GEI's and to run all of them a faster processor id needed. ResNet was trained on Casia-B dataset. Casia-B dataset has various defined experimental protocols. For a fair comparison I follow the protocols in the baseline methods. The protocol followed, uses first 74 subjects for training the model and rest 50 subjects for testing. The parameters of ResNet were first trained on all the GEIs of first 74 subjects. Once the model was trained, gallery and probe were set for testing it on rest

of 50 subjects. Gallery consists of GEIs of normal walk 1 to 4 at all viewing angles of each of the remaining 50 subjects. The pre-trained model was further trained on gallery of remaining subjects. For validation or detection, the probe was set to bag walk 1-2, clothes walk 1-2 and normal walk 5-6 at each individual viewing angle of subjects and then accuracies were compared. Surprisingly ResNet surpass the state-of-the art methods with huge margins for normal walk at every angle and bag and clothes walk at most of the angles of Casia-B dataset. After extensive training and testing on Casia-B dataset I used it in my system using a borrowed processor to get the results.

3.6 Implementation Of Resnet

Fastai is a framework that sits on top of pytorch. It simplifies the process of training fast and accurate neural network using best modern practices. Following code explains the implementation of ResNet using fastai framework.

```
In [ ]: from fastai.vision import *  
  
In [ ]: path = Path('data74')  
  
In [ ]: classes = ["{:03}".format(i) for i in range(1, 75)]  
        print (classes)
```

These lines of code show the standard way of importing fastai library for further use. Path is set to the folder “data74” which contains the data of first 74 subjects for training ResNet34. Furthermore, the classes are defined from 001 to 074 as there are 74 subjects. The data74 folder contains 74 folders from 000 to 074 each containing 110 Geis of subjects.

```
In [ ]: for c in classes:
        print(c)
        verify_images(path/c, delete=True, max_size=240)
```

```
In [1]: np.random.seed(42)
        data=ImageDataBunch.from_folder(path,train=".",valid_pct = 0.2,ds_tfms=get_transforms(),size=(320,240),num_workers=4)
        |.normalize(imagenet_stats)
```

The Geis are further verified for a corrupt image as it is not possible to train a neural network with corrupt data. To build a deep learning model I need annotated data with correct labels. In fastai there is an object named as ImageDataBunch which makes my work easy by sorting the data according to the labels and creates a data bunch or data object on which my neural network can be trained. So, the data is divided into training and validation set. `valid_pct = 0.2` makes sure that the validation set is the 20 percent of all the examples present in the data set. The neural network will never see this validation set while training itself on the training set. After training, the neural network is tested on this validation set for its accuracy.

```
In [ ]: learn = cnn_learner(data, models.resnet34, metrics = (error_rate, accuracy))
```

Next, I load a pre-trained ResNet34 model on ImageNet database. The parameters of ResNet34 are trained on ImageNet dataset but we want ResNet to classify Geis so we train the ResNet34 model according to my need by next unfreezing the parameters so that we can train every layer of ResNet on Casia-B dataset.

```
In [ ]: learn.unfreeze()
```

```
In [ ]: learn.lr_find()
```

```
In [ ]: learn.recorder.plot()
```

After unfreezing the parameters, I first plot the learning rate so that before training ResNet34 on Casia-B I can manually decide the learning rate. Once the learning rate is decided I am good to train the model.

```
In [ ]: learn.fit_one_cycle(30, max_lr=slice(3e-04, 3e-02))
```

The function `fit_one_cycle()` of learner object passes two arguments. One is the number of epochs and second is the learning rate and this starts the training of our model on training set and check its accuracy on new examples from valid set. Once the model is trained and validated, I need to show the results in quantitative manner.

```
In [ ]: interp = ClassificationInterpretation.from_learner(learn)
```

```
In [ ]: interp.plot_confusion_matrix(figsize=(200,200))
```

```
In [ ]: interp.plot_top_losses(9, figsize = (20,20))
```

ClassificationInterpretation object allows us to see the results in a quantitative manner. It accepts a learner object as an argument. Once the object is created, I can see the confusion matrix and top losses by calling the functions given in the code above.

```
In [ ]: learn.export('data74')
```

The next step is to export this trained model on gait energy images so that I can further train and validate it on the remaining 50 Casia-B subjects. For this purpose, export function is called of the learner object so that the trained model is saved on disk. In this case I have named my trained model as data74 since it was trained on 74 subjects.

3.7 Gallery Probe Testing

Once ResNet34 is trained on 74 subjects following the protocol it is then further trained and tested on remaining 50 subjects by dividing the training and validation set in gallery and probe.

```
In [8]: learn1 = load_learner('D:/FYP/FYP/models', 'data74')
learn1.data = data
```

The pretrained data74 ResNet model is loaded into a new learner object and data bunch is created in which training set is the gallery and validation sets are probes as defined by the protocol.

```
In [9]: learn1.fit_one_cycle(30)
```

I further train the pretrained model on gallery.

```
learn = load_learner(path)
folder = ["{0:03}".format(i) for i in range(75, 125)]
pic = ['nm-05', 'nm-06']
count = 0
for i in folder:
    for j in pic:
        img = open_image('C:/Users/Nauman Riaz/Desktop/FYP/register/valid/'+i+'/'+i+'_'+j+'_090.jpg')
        pred_class, pred_idx, outputs = learn.predict(img)
        # print(pred_class)
        if str(pred_class) == i:
            count = count + 1
        else:
            print(str(pred_class) + '=' + i)
print (count)
```

The code above shows the final testing phase where the model is probed over validation sets defined in the protocol to determine the accuracy of the model. The results of ResNet on Casia-B are showed in chapter 3 and compared with other models on gait recognition. ResNet beats state of the art in most cases and can be easily implemented for real world gait recognition.

3.8 Results and Discussion

The following table shows the resulting accuracies obtained according to the explained experiments and distribution of data into training and testing sets. The results are compared with some of the state-of-the-arts papers and their accuracies are reported directly from their papers, which can be found in the reference chapter. After finding accuracies on each angle, the results are also averaged on all the angles excluding identical views.

Table 3.1 - Accuracies on CASIA-B, excluding identical views

Gallery NM#1-4	0° - 180°											Mean
Probe	0°	18°	36°	54°	72°	90°	108°	126°	144°	162°	180°	

NM # 5-6	CNN-3D[23]	87.1	93.2	97.0	94.6	90.2	88.3	91.1	93.8	96.5	96.0	85.7	92.1
	CNN	88.7	95.1	98.2	96.4	94.1	91.5	93.9	97.5	98.4	95.8	85.6	94.1
	GaitSet	90.8	97.9	99.4	96.9	93.6	91.7	95.0	97.8	98.9	96.8	85.8	95.0
	ResNet(mine)	100	100	99	99	99	100	100	100	100	100	100	99.72

BG # 1-2	CNN-LB	64.2	80.6	82.7	76.9	64.8	63.1	68.0	76.9	82.2	75.4	61.3	72.4
	GaitSet	83.8	91.2	91.8	88.8	83.3	81.0	84.1	90.0	92.2	94.4	79.0	87.2
	ResNet(mine)	91	86	79	75	72	60	70	75	72	83	83	76.9

The table clearly shows that my proposed model obtains very high frequencies on NM dataset, with only angle 36°, 54° and 72° at 99% while all the rest showing 100% accuracy, beating the state-of-the-art methods mean accuracies by 4.72% at 99.72%. In other 3 papers, it can be observed that 90° gives a local minimum value, always less than

72° and 108°, the possible reason for which is given in GaitSet paper. But my proposed model overcomes this barrier too by achieving 100% at 90°.

However, in terms of BG subsets, my model performs better than one state-of-the-art model, exceeding the mean accuracy by 4.5% but lagging behind the other state-of-the-art method by a mean accuracy of 10.3%.

CHAPTER 4. LABOUR PROTECTION AND SAFETY IN EMERGENCY

4.1 Labour Protection

What is an emergency action plan?

Employers and employees must follow specific procedures outlined in an emergency action plan to protect workers from fires and other calamities. Not every employer is required to set up an emergency plan of action. Even if it is not explicitly stated that you must. Consequently, creating an emergency action plan is a wise strategy to safeguard. During an emergency, protect your business, your employees, and yourself. Creating a thorough emergency action plan that addresses dealing with all kinds of problems unique to your workplace is not tough. It can be advantageous to include your management team and workers involved in the process. Outline how you seek to protect lives and property in the case of natural catastrophic.

It is simple to put together a thorough emergency action plan that addresses all problems unique to your workplace.

It could be advantageous to involve your management group and staff in the process. Ask for their assistance in creating and putting into action your emergency action plan as you describe your desire to safeguard people and property in the case of an emergency. Their cooperation and backing are essential to the strategy's success.

What should be included in your emergency action plan?

A wide range of potential situations that can arise at work should be considered while creating your emergency action plan. It needs to be personalized for your workplace and contain details on any potential emergency sources. Performing a hazard assessment to identify any potential physical or chemical risks in your workplaces will help you develop an emergency action plan. If you have multiple work locations, each one needs to have an emergency action plan.

Your emergency action plan must at the very least contain the following items:

1. An evacuation policy and procedure;

2. A preferred means for reporting fires and other emergencies;
3. Emergency escape routes assigned, such as floor layouts, workplace maps, and safe or refuge zones;
4. Procedures for employees who must stay to perform or shut down critical plant operations, operate fire extinguishers, or perform other essential services that cannot be shut down for every emergency alarm before evacuating;
5. Names, titles, departments, and telephone numbers of people both inside and outside your company to contact for additional information or an explanation of duties and responsibilities under the emergency plan;
6. Rescue and medical duties.

Additionally, you might want to think about establishing assembly guidelines and a spot to gather everyone after an evacuation.

The location of an alternative communications center to be used in the event of a fire or explosion, as well as a secure on- or offsite location to store originals or duplicate copies of your accounting records, legal documents, employee emergency contact lists, and other crucial records are also things you might find useful to include in your plan.

How can an emergency be announced to the staff?

Your plan must specify how to notify staff members, including disabled personnel, of the need to evacuate or take other action, as well as how to report crises as needed. You need to take a number of steps, including:

1. Ensure that alarms are unique and understood by all workers as a signal to leave the work area or carry out the measures specified in your plan;
2. A public address system, a portable radio, or other emergency communication device should be made available to employees so they can learn about the emergency and contact local law enforcement, the fire department, and others.
3. You should also specify that all employees in the workplace must be able to hear, see, or otherwise perceive alarms. If the electricity goes out, you might want to think about establishing a backup power source.

You also might want to think about the following:

1. Using tactile devices to alert workers who would not otherwise be able to recognize an audible or visual alarm;
2. Providing an updated list of important personnel, such as the plant manager or physician, to notify in the event of an emergency during off-duty hours, in order of priority.

How are evacuation policies and procedures created?

Confusion, injuries, and property damage may occur during a chaotic evacuation. This is why it's crucial to consider the following while creating your emergency action plan:

1. Conditions that would necessitate an evacuation include: A clear chain of command and identification of the person in your charge business with the power to impose a closure or evacuation.
2. You could appoint an "evacuation warden" to help others in an evacuation and to keep troops in the loop;
3. Particular evacuation protocols, including exits and routes. Post these guidelines in a location where all staff members can easily view them;
4. Procedures for helping those who are disabled or do not understand English;
5. The designation of which personnel, if any, will continue or halt important operations during an evacuation. These individuals must be capable of knowing when to give up on the mission and leave on their own;
6. A mechanism for tracking down persons after an evacuation. Think about the transportation requirements of the staff during community-wide evacuations.

What circumstances need a request for an evacuation?

Local emergency personnel may issue an evacuation order in the event of an emergency. They might give you instructions to turn off the electricity, gas, and water in some circumstances. Listen to newscasts if you have access to radio or television to stay informed and obey any directives from the authorities.

In other situations, a designated individual within your company should be in charge of deciding whether to evacuate or halt operations. Everyone in the building

should have their health and safety as their top priority. The easiest strategy to protect personnel during a fire is to immediately evacuate to a location outside of the building. On the other hand, in some situations—such as a poisonous gas release at a plant across town from your place of business—evacuating employees might not be the wisest course of action.

Your choice could be influenced by the type of building you work in.

The majority of structures are susceptible to the effects of natural calamities like tornadoes, earthquakes, floods, or explosions. The sort of emergency and the structure of the building both affect how much damage is done. For instance, modern manufacturing and office buildings have steel frames and are more structurally sound than nearby commercial properties. But in a catastrophe like a significant earthquake or explosion, almost every kind of structure will be impacted. Some buildings will fall, while others will have walls and floors that are less sturdy.

What function do emergency coordinators and evacuation wardens serve?

You might want to choose a trustworthy person to organize your emergency plan and evacuation when you're writing your emergency action plan. Employees must be aware of the coordinator's identity and understand that they have the power to make decisions in an emergency.

The coordinator should be in charge of: Assessing the situation to see if an emergency exists that necessitates the use of your emergency procedures;

1. Directing all activities in the region, including personnel evacuation;
2. Coordinating emergency services from outside, such medical assistance and neighborhood fire departments, making sure they're accessible and informed as required; and
3. Making the necessary decisions to shut down plant operations.

You could choose to appoint evacuation wardens in addition to a coordinator to assist in moving staff members during an emergency from dangerous regions to safe ones. In general, there should be one warden for every 20 workers, and the right number of wardens should be on duty always during business hours.

Employees assigned to help with emergency evacuation procedures should receive training on the layout of the entire workplace and different alternate escape routes. Employees with special needs who might need extra support, the buddy system, and dangerous places to avoid during an emergency evacuation should all be made aware of to all employees and those assigned to help in crises.

How are escape routes and exits established?

Determine primary and secondary escape routes and exits while creating your emergency action plan. As much as it is possible given the circumstances, make sure that emergency exits and evacuation routes are:

1. Clearly marked and well lit;
2. Wide enough to accommodate the number of evacuating personnel;
3. Unlikely to expose evacuating personnel to additional hazards.

Unobstructed and clear of debris at all times; and

If you create diagrams outlining escape routes and exits, display them clearly where all employees can see them.

4.2 Emergency safety

Major mishaps and catastrophes can result in fatalities and considerable property loss. As a result, maintaining trouble-free operation at the company should be viewed as a crucial duty that demands the focus of engineers and technical staff in addition to managers at all levels. Natural disasters, errors in enterprise design, construction, and equipment, the start-up of facilities with significant flaws and deviations from plans, the commissioning of ventilation systems without testing them for effectiveness, safety and labour protection issues, inadequate equipment with control and measuring, protective, and blocking equipment, and inadequate tightness of technological eq They might also be caused by faulty wiring, technological procedures, or a lack of effective fire extinguishing equipment.

Because of staff members' poor awareness, ignorance, and violations of safety regulations, each single accident is the consequence of a complex interaction of causes and unfavourable variables. The research of accident causes and a thorough evaluation of the level of danger enable accurate determination of preventive actions, the provision of required safeguards for people's safety, and the minimization of damage.

The following are the primary steps taken to minimise the effects of major accidents:

1. Warning of the danger to workers and employees.
2. Thorough reconnaissance of the accident site.
3. Rescue from rubble.
4. Treating victims and transporting them to medical facilities.
5. Firefighting.
6. Localization of accidents on utility and energy networks that obstruct rescue operations.
7. Arrangement of emergency response systems; and

Significant forces and resources are needed for quick rescue operations and the swift eradication of the accident's effects; for these goals, special and territorial units with a general mission are involved.

To ensure the continuation of work at an increasing rate, rescue operations at the accident scene are conducted under gassy conditions, and in the case of flames, smoky and high temperatures. Resources are divided into changes and allocated reserves.

Immediately following the catastrophe, rescue efforts are coordinated, as well as support for casualties. The first responders at the scene of an accident must be firefighters, police officers, and ambulances.

Victims are extricated from the wreckage and debris and given first aid and medical attention if they are in a condition of shock. People are rescued from significant obstructions in accordance with safety precautions, given emergency medical care, and then transported to medical facilities.

To coordinate disaster and disaster relief, a permanent emergency task force headed by the Chief Engineer is being established at the location. It operates in times of disaster under the general supervision of the district (city) emergency commission.

CONCLUSION

In this report, I tackled the gait recognition problem using a new implementation of convolutional neural network ResNet34 provided by the famous python library Fastai, written by Jeremy Howard. For extracting features and properties from gait data recorded by the camera, I used traditional algorithms and techniques used in most of the gait recognition systems. However, to further improve the results, I developed some of my own algorithms specific to the data type and applied them as well. The results obtained clearly indicates that this new implementation of ResNet34 by Fastai is really promising in the field of gait recognition. Along with the research, I also created a full-fledged working prototype of our gait recognition system which is ready to be deployed in the market.

My research has opened new doors for the scientists in this field. However, there is still a lot of work that can be done on my proposed system to make it the best in the world. For example, I did not perform any image processing on the final GEIs or the data collected from university to be tested that all part went into research, doing which I believe will strongly enhance the final accuracy of the system and it might be able to outperform state-of-the-art GaitSet in winter clothes and carrying a bag walking variation.

REFERENCES

1. Niyogi and Adelson, Analyzing and recognizing walking figures in XYT, <https://ieeexplore.ieee.org/document/323868>
2. S. Sarkar, The humanID gait challenge problem: Data sets, performance, and analysis, <https://ieeexplore.ieee.org/document/1374864>
3. David Cunado and Mark Nixon, Using gait as a biometric, via phase-weighted magnitude spectra, <https://link.springer.com/chapter/10.1007/BFb0015984>
4. Jang-Hee Yoo & Mark Nixon, Markerless human gait analysis via image sequences, https://www.researchgate.net/publication/37536578_Markerless_Human_Gait_Analysis_via_Image_Sequences
5. Raquel and Fua, 3D tracking for gait characterization and recognition, https://www.researchgate.net/publication/2605293_3D_Tracking_for_Gait_Characterization_and_Recognition
6. J. Han and Bir Bhanu, Individual recognition using gait energy image, <https://ieeexplore.ieee.org/document/1561189>
7. Liu and Zheng, Gait history image: A novel temporal template for gait recognition, <https://ieeexplore.ieee.org/document/4284737>
8. Heikki Ailisto, Identifying people from gait pattern with accelerometers, https://www.researchgate.net/publication/241529587_Identifying_people_from_gait_pattern_with_accelerometers
9. Micheal Otero, Application of a continuous wave radar for human gait recognition, <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5809/0000/Application-of-a-continuous-wave-radar-for-human-gait-recognition/10.1117/12.607176.short?SSO=1>
10. K. Nakajima, Footprint-based personal recognition, <https://ieeexplore.ieee.org/document/880106>

11. Hanqing Chao, Yiwei He, Junping Zhang and Jianfeng Feng, GaitSet: Regarding gait as a set for cross-view gait recognition, <https://arxiv.org/abs/1811.06186>
12. Yuqi Zhang, Yongzhen Huang, Liang Wang and Shiqi Yu, A comprehensive study on gait biometrics using a joint CNN-based method, <https://www.sciencedirect.com/science/article/abs/pii/S0031320319301694>
13. CBSR, CASIA Dataset, <http://www.cbsr.ia.ac.cn/english/Gait%20Databases.asp>
14. Carnegie Mellon University, CMU Dataset, <http://mocap.cs.cmu.edu/>
15. Osaka University, OU-MVLP Database, <http://www.am.sanken.osaka-u.ac.jp/BiometricDB/GaitLP.html>
16. Adrian Rosebrock. Pedestrian Detection Opencv. <https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/>
17. OpenCV, Background Subtraction, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html
18. Z. Zivkovic, Improved adaptive Gaussian mixture model for background subtraction, <https://ieeexplore.ieee.org/document/1333992>
19. Z. Zivkovic, Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction, <https://www.sciencedirect.com/science/article/abs/pii/S0167865505003521>
20. OpenCV, Morphological Transformations, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html
21. PyQt5, <https://pypi.org/project/PyQt5/>
22. Fastai, Jeremy Howard, <https://www.fast.ai/>
23. Zifeng Wu, Yongzhen Huang, Liang Wang, Xiaogang Wang, Tieniu, A Comprehensive Study on Cross-View Gait Based Human Identification with Deep CNNs, <https://ieeexplore.ieee.org/document/7439821>

Appendix A

CODES

1. PRE TREATMENT:

Importing libraries

In [1]:

```
import PIL
```

In [2]:

```
import scipy
```

In [3]:

```
import os
from scipy import misc as scisc
import cv2
import numpy as np
from warnings import warn
from time import sleep
import argparse
import imageio
from multiprocessing import Pool
from multiprocessing import TimeoutError as MP_TimeoutError
```

Defining constants

In [4]:

```
START = "START"
FINISH = "FINISH"
WARNING = "WARNING"
FAIL = "FAIL"
```

#Height and width of image after pretreatment

```
T_H = 64
T_W = 64
```

In [5]:

```
INPUT_PATH = "/content/drive/MyDrive/CASIA-B"
OUTPUT_PATH = "/content/drive/MyDrive/Pretreatment Output"
IF_LOG = True
LOG_PATH = "/content/drive/MyDrive/pretreatment.log"
WORKERS = 1
```

Defining functions

In [6]:

```
def boolean_string(s):
```

```

if s.upper() not in {'FALSE', 'TRUE'}:
    raise ValueError('Not a valid boolean string')
return s.upper() == 'TRUE'

```

In [7]:

```

def log2str(pid, comment, logs):
    str_log = ""
    if type(logs) is str:
        logs = [logs]
    for log in logs:
        str_log += "# JOB %d: --%s-- %s\n" % (
            pid, comment, log)
    return str_log

```

In [8]:

```

def log_print(pid, comment, logs):
    str_log = log2str(pid, comment, logs)
    if pid != 0:
        print("\n" + str_log)
    if comment in [WARNING, FAIL]:
        with open(LOG_PATH, 'a') as log_f:
            log_f.write(str_log)
    if comment in [START, FINISH]:
        if pid % 500 != 0:
            return
    print(str_log, end="")

```

In [9]:

```

def cut_img(img, seq_info, frame_name, pid):
    # A silhouette contains too little white pixels
    # might be not valid for identification.
    if img.sum() <= 10000:
        message = 'seq:%s, frame:%s, no data, %d.' % (
            '.'.join(seq_info), frame_name, img.sum())
        warn(message)
        log_print(pid, WARNING, message)
        return None
    # Get the top and bottom point
    y = img.sum(axis=1)
    y_top = (y != 0).argmax(axis=0)
    y_btm = (y != 0).cumsum(axis=0).argmax(axis=0)
    img = img[y_top:y_btm + 1, :]
    # As the height of a person is larger than the width,
    # use the height to calculate resize ratio.
    _r = img.shape[1] / img.shape[0]
    _t_w = int(T_H * _r)
    img = cv2.resize(img, (_t_w, T_H), interpolation=cv2.INTER_CUBIC)
    # Get the median of x axis and regard it as the x center of the person.

```

```

sum_point = img.sum()
sum_column = img.sum(axis=0).cumsum()
x_center = -1
for i in range(sum_column.size):
    if sum_column[i] > sum_point / 2:
        x_center = i
        break
if x_center < 0:
    message = 'seq:%s, frame:%s, no center.' % (
        '-'.join(seq_info), frame_name)
    warn(message)
    log_print(pid, WARNING, message)
    return None
h_T_W = int(T_W / 2)
left = x_center - h_T_W
right = x_center + h_T_W
if left <= 0 or right >= img.shape[1]:
    left += h_T_W
    right += h_T_W
    _ = np.zeros((img.shape[0], h_T_W))
    img = np.concatenate([_, img, _], axis=1)
img = img[:, left:right]
return img.astype('uint8')

```

In [10]:

```

def cut_pickle(seq_info, pid):
    seq_name = '-'.join(seq_info)
    log_print(pid, START, seq_name)
    seq_path = os.path.join(INPUT_PATH, *seq_info)
    out_dir = os.path.join(OUTPUT_PATH, *seq_info)
    frame_list = os.listdir(seq_path)
    frame_list.sort()
    count_frame = 0
    for _frame_name in frame_list:
        frame_path = os.path.join(seq_path, _frame_name)
        img = cv2.imread(frame_path)[:, :, 0]
        img = cut_img(img, seq_info, _frame_name, pid)
        if img is not None:
            # Save the cut img
            save_path = os.path.join(out_dir, _frame_name)
            imageio.imwrite(save_path, img)
            count_frame += 1
    # Warn if the sequence contains less than 5 frames
    print(count_frame)
    if count_frame < 5:
        message = 'seq:%s, less than 5 valid data.' % (
            '-'.join(seq_info))
        warn(message)

```

```
log_print(pid, WARNING, message)
```

```
log_print(pid, FINISH,  
          'Contain %d valid frames. Saved to %s.'  
          % (count_frame, out_dir))
```

Main body

In [11]:

```
pool = Pool(WORKERS)  
results = list()  
pid = 0  
  
print('Pretreatment Start.\n'  
      'Input path: %s\n'  
      'Output path: %s\n'  
      'Log file: %s\n'  
      'Worker num: %d' % (  
          INPUT_PATH, OUTPUT_PATH, LOG_PATH, WORKERS))  
  
id_list = os.listdir(INPUT_PATH)  
id_list.sort()  
  
# Walk the input path  
for _id in id_list:  
    seq_type = os.listdir(os.path.join(INPUT_PATH, _id))  
    seq_type.sort()  
    for _seq_type in seq_type:  
        view = os.listdir(os.path.join(INPUT_PATH, _id, _seq_type))  
        view.sort()  
        for _view in view:  
            seq_info = [_id, _seq_type, _view]  
            out_dir = os.path.join(OUTPUT_PATH, *seq_info)  
            os.makedirs(out_dir)  
            results.append(pool.apply_async(cut_pickle(seq_info, pid)))  
            sleep(0.02)  
            pid += 1  
  
pool.close()  
unfinish = 1  
while unfinish > 0:  
    unfinish = 0  
    for i, res in enumerate(results):  
        try:  
            res.get(timeout=0.1)  
        except Exception as e:  
            if type(e) == MP_TimeoutError:  
                unfinish += 1
```

```
        continue
    else:
        print('\nERROR OCCUR: PID %d, ERRORTYPE: %s\n',
              i, type(e))
        #raise e
pool.join()
```

```
from google.colab import drive
drive.mount('/content/drive')
```

2. GUI

Make sure you have install the following Python packages, we recommend you to install Anaconda and OpenCV 2.4. This code can work in both Windows and Linux.

```
import sys

import cPickle

from PyQt4 import QtCore, QtGui, uic

import numpy as n

from PIL import Image

from PIL.ImageQt import ImageQt

import cv2

import os

#Loading the UI window

qtCreatorFile = "GaitUI.ui"

Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)

def pickle(filename, data, compress=False):

    fo = open(filename, "wb")

    cPickle.dump(data, fo, protocol=cPickle.HIGHEST_PROTOCOL)

    fo.close()
```

```
def unpickle(filename):
```

```
    fo = open(filename, 'rb')
```

```
    dict = cPickle.load(fo)
```

```
    fo.close()
```

```
    return dict
```

```
class GaitDemo(QtGui.QMainWindow, Ui_MainWindow):
```

```
    def __init__(self):
```

```
        """
```

```
        Set initial parameters here.
```

```
        Note that the demo window size is 1366*768, you can edit this via Qtcreator.
```

```
        In this demo, we take 20 frames of profiles to generate a GEI. You can edit this number by your  
self.
```

```
        """
```

```
        QtGui.QMainWindow.__init__(self)
```

```
        Ui_MainWindow.__init__(self)
```

```
        self.showFullScreen()
```

```
        self.setupUi(self)
```

```
        self.capture = cv2.VideoCapture(0)# Edit this default num to 1 or 2, if you have multiple cameras.
```

```
        self.currentFrame=n.array([])
```

```
        self.firstFrame=None
```

```
        self.register_state = False
```

```
        self.recognition_state = False
```

```
        self.save_on = False
```

```
        self.gei_fix_num = 20
```

```
        #Ok, this is our logos, welcome to contact and visit us.
```

```
        self.logo1 = QtGui.QLabel(self.centralwidget)
```

```
        self.logo1.setPixmap(QtGui.QPixmap(QtGui.QPixmap('logo1.png').scaled(250,70)))
```

```
        self.logo1.setGeometry(5, 698, 250, 70)
```

```
        self.logo2 = QtGui.QLabel(self.centralwidget)
```

```

self.logo2.setPixmap(QtGui.QPixmap(QtGui.QPixmap('logo2.png').scaled(170,70)))
self.logo2.setGeometry(275, 698, 170, 70)
self.logo3 = QtGui.QLabel(self.centralwidget)
self.logo3.setPixmap(QtGui.QPixmap(QtGui.QPixmap('logo3.png').scaled(243,70)))
self.logo3.setGeometry(480, 698, 243, 70)

#Set two window for raw video and segmentation.
self.video_label=QtGui.QLabel(self.centralwidget)
self.seg_label = QtGui.QLabel(self.centralwidget)
self._timer = QtCore.QTimer(self)
self.video_label.setGeometry(50,100, 512, 384)
self.seg_label.setGeometry(612,100, 512, 384)
self.load_dataset()
self._timer.timeout.connect(self.play)

#Waiting for you to push the button.
self.register_2.clicked.connect(self.register_show)
self.recognize.clicked.connect(self.recognition_show)
self.updater.clicked.connect(self.update_bk)
self.save_gei.clicked.connect(self.save_gei_f)
self._timer.start(27)
self.update()

def save_gei_f(self):
    """
    Waiting the save button.
    """
    self.save_on = True
    self.state_print.setText('Saving!')

```

```

def register_show(self):
    """
    To record the GEI into gait database.
    """
    self.register_state = True
    self.recognition_state = False
    self.state_print.setText('Register!')
    self.gei_current = n.zeros((128,88), n.single)
    self.numInGEI = 0

def load_dataset(self):
    """
    Load gait database if existing.
    """
    self.data_path = './GaitData'
    if os.path.exists(self.data_path):
        dic = unpickle(self.data_path)
        self.num = dic['num']
        self.gei = dic['gei']
        self.name = dic['name']
    else:
        self.num = 0
        self.gei = n.zeros([100,128,88],n.uint8)
        self.name = []
        dic = {'num':self.num, 'gei':self.gei, 'name':self.name}
        pickle(self.data_path, dic, compress=False)
    self.id_num.setText('%d' %self.num)
    self.state_print.setText('Running!')

def recognition_show(self):

```



```

'''
Working now and just recognizing the one in front of this camera.
'''

self.recognition_state = True
self.register_state = False
self.gei_current = n.zeros((128,88), n.single)
self.numInGEI = 0
self.state_print.setText('Recognition!')

def update_bk(self):
    '''
    If you moved the camera.
    '''
    self.firstFrame = self.FrameForUpdate

def play(self):
    '''
    Main program.
    '''
    ret, frame=self.capture.read() #Read video from a camera.
    if(ret==True):
        frame = cv2.resize(frame,(512,384))
        #Apply background subtraction method.
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (3,3), 0)
        if self.firstFrame is None:
            self.firstFrame = gray # Set this frame as the background.
        frameDelta = cv2.absdiff(self.firstFrame, gray)
        thresh = cv2.threshold(frameDelta, 50, 255, cv2.THRESH_BINARY)[1]
        self.FrameForUpdate = gray

```

```

thresh = cv2.dilate(thresh, None, iterations=2)
(cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
thresh = n.array(thresh)
max_rec=0

#Find the max box.
for c in cnts:
    if cv2.contourArea(c) < 500:
        continue
    (x, y, w, h) = cv2.boundingRect(c)

    if w>25 and h>50:
        if max_rec<w*h:
            max_rec = w*h
            (x_max, y_max, w_max, h_max) = cv2.boundingRect(c)
#If exist max box.
if max_rec>0:
    cv2.rectangle(frame, (x_max, y_max), (x_max + w_max, y_max + h_max), (0, 255, 0), 2)
    if x_max>20: #To ignore some regions which contain parts of human body.
        if self.register_state or self.recognition_state:
            nim = n.zeros([thresh.shape[0]+10,thresh.shape[1]+10],n.single) # Enlarge the box for
better result.
            nim[y_max+5:(y_max + h_max+5),x_max+5:(x_max + w_max+5)] =
thresh[y_max:(y_max + h_max),x_max:(x_max + w_max)]
            offsetX = 20
            # Get coordinate position.
            ty, tx = (nim >100).nonzero()
            sy, ey = ty.min(), ty.max()+1
            sx, ex = tx.min(), tx.max()+1
            h = ey - sy

```

```
w = ex - sx
```

```
if h>w:# Normal human should be like this, the height should be greater than width.
```

```
    # Calculate the frame for GEI
```

```
    cx = int(tx.mean())
```

```
    cenX = h/2
```

```
    start_w = (h-w)/2
```

```
    if max(cx-sx,ex-cx)<cenX:
```

```
        start_w = cenX - (cx-sx)
```

```
    tim = n.zeros((h,h), n.single)
```

```
    tim[:,start_w:start_w+w] = nim[sy:ey,sx:ex]
```

```
    rim = Image.fromarray(n.uint8(tim)).resize((128,128), Image.ANTIALIAS)
```

```
    tim = n.array(rim)[:,offsetX:offsetX+88]
```

```
    if self.numInGEI<self.gei_fix_num:
```

```
        self.gei_current += tim # Add up until reaching the fix number.
```

```
    self.numInGEI += 1
```

```
if self.numInGEI>self.gei_fix_num:
```

```
    if self.save_on:
```

```
        #Save the GEI.
```

```
        self.gei[self.num,,:] = self.gei_current/self.gei_fix_num
```

```
Image.fromarray(n.uint8(self.gei_current/self.gei_fix_num)).save('./gei/gei%02d%s.jpg'%(self.num,self.id_name.toPlainText()))
```

```
    self.name.append(self.id_name.toPlainText())
```

```
    self.num +=1
```

```
    self.id_num.setText('%d' %self.num)
```

```
    dic = {'num':self.num, 'gei':self.gei, 'name':self.name }
```

```
    pickle(self.data_path, dic, compress=False)
```

```
    self.save_on = False
```

```
    self.state_print.setText('Saved!')
```

```
elif self.recognition_state:
```

```

#Recognition.
self.gei_query = self.gei_current/(self.gei_fix_num)
score = n.zeros(self.num)
self.gei_to_com = n.zeros([128,88],n.single)
for q in xrange(self.num):
    self.gei_to_com = self.gei[q,:,:]
    score[q]=n.exp(-(((self.gei_query[:]-
self.gei_to_com[:])/(128*88))*2).sum())#Compare with gait database.
    q_id = score.argmax()
    if True:
        id_rec = '%s' %self.name[q_id]

cv2.putText(frame,id_rec,(x_max+20,y_max+20),fontFace=cv2.FONT_HERSHEY_SIMPLEX,
fontScale=1.0, thickness=2,color=(0,0,255))

else:
    self.gei_current = n.zeros((128,88), n.single)
    self.numInGEI = 0

#Show results.
self.currentFrame=cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
self.currentSeg=Image.fromarray(thresh).convert('RGB')
self.currentSeg = ImageQt(self.currentSeg)
height,width=self.currentFrame.shape[:2]
img=QtGui.QImage(self.currentFrame,
width,
height,
QtGui.QImage.Format_RGB888)
img=QtGui.QPixmap.fromImage(img)
self.video_label.setPixmap(img)
seg=QtGui.QImage(self.currentSeg)
seg=QtGui.QPixmap(seg)

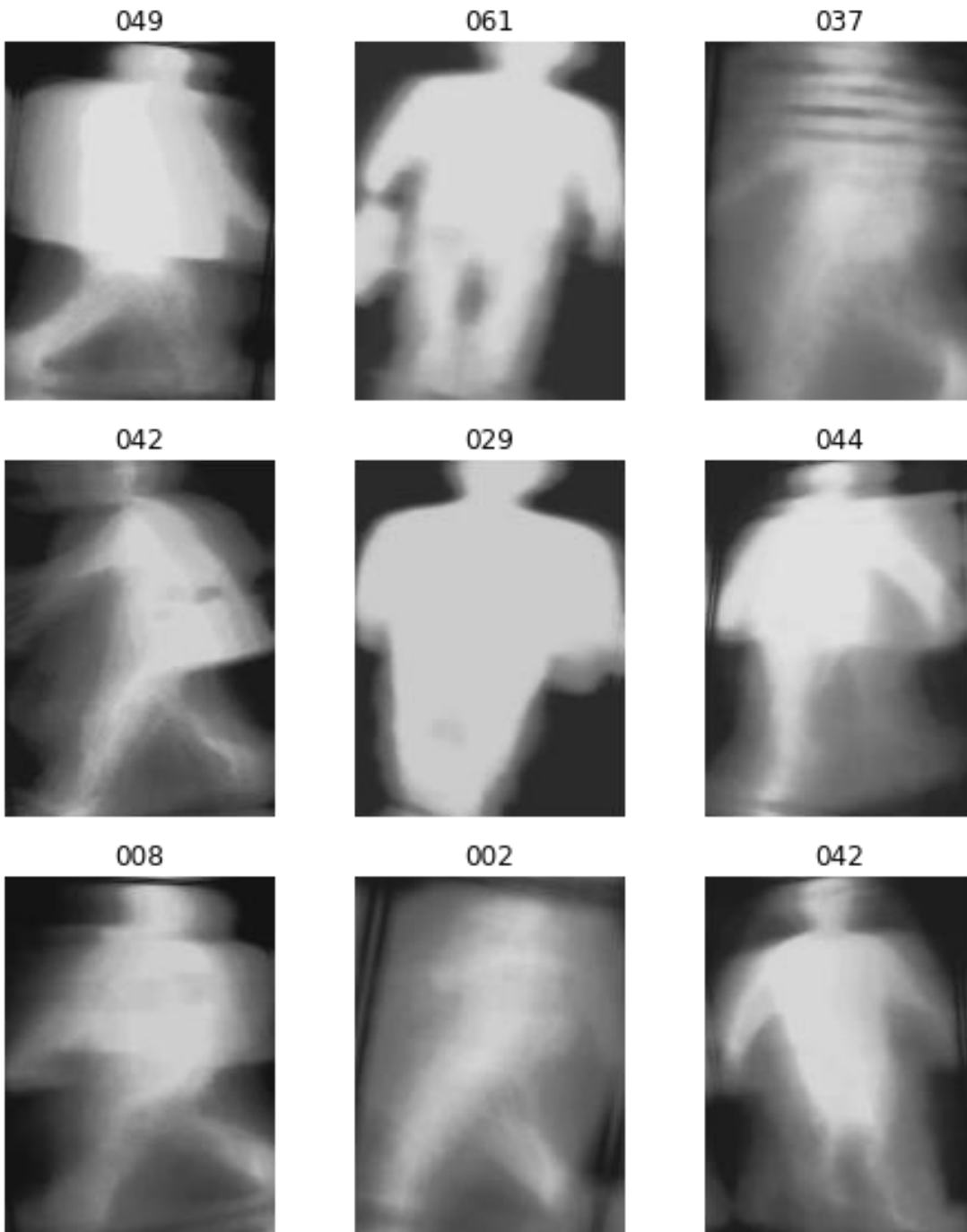
```

```
self.seg_label.setPixmap(seg)
```

```
if __name__ == "__main__":  
    app = QtGui.QApplication(sys.argv)  
    window = GaitDemo()  
    window.show()  
    sys.exit(app.exec_())
```

3. Data74

```
from fastai.vision import *  
path = Path('data74')  
classes = ["{0:03}".format(i) for i in range(1, 75)]  
print (classes)  
['001', '002', '003', '004', '005', '006', '007', '008', '009', '010', '011', '012',  
'013', '014', '015', '016', '017', '018', '019', '020', '021', '022', '023', '024',  
'025', '026', '027', '028', '029', '030', '031', '032', '033', '034', '035', '036',  
'037', '038', '039', '040', '041', '042', '043', '044', '045', '046', '047', '048',  
'049', '050', '051', '052', '053', '054', '055', '056', '057', '058', '059', '060',  
'061', '062', '063', '064', '065', '066', '067', '068', '069', '070', '071', '072',  
'073', '074']  
for c in classes:  
    print(c)  
    verify_images(path/c, delete=True, max_size=500)  
doc(ImageDataBunch.from_folder)  
np.random.seed(42)  
data = ImageDataBunch.from_folder(path, train=".", valid_pct = 0.2, ds_tfms = get_transforms(),  
size=(320,240), num_workers=4).normalize(imagenet_stats)  
data.classes  
data.show_batch(rows=3, figsize=(7,8))
```



```
data.classes, data.c, len(data.train_ds), len(data.valid_ds)
```

```
learn = cnn_learner(data, models.resnet34, metrics = (error_rate, accuracy))
```

```
learn.fit_one_cycle(15)
```

```
learn.save('stage-1')
```

```
learn.unfreeze()
```

```
learn.lr_find()
```

```
learn.recorder.plot()
```

```

learn.save('stage-2')
learn.load('stage-2')
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix(figsize=(200,200))
interp.plot_top_losses(9, figsize = (20,20))
from fastai.widgets import *
db = (ImageList.from_folder(path)
      .split_none()
      .label_from_folder()
      .transform(get_transforms(), size=224)
      .databunch()
      )
learn_cln = cnn_learner(db, models.resnet34, metrics=error_rate)

learn_cln.load('stage-2');
ds, idxs = DatasetFormatter().from_toplosses(learn_cln)
ImageCleaner(ds, idxs, path)
ds, idxs = DatasetFormatter().from_similars(learn_cln)
ImageCleaner(ds, idxs, path, duplicates=True)
learn.export('data74')
defaults.device = torch.device('cpu')
img = open_image('C:/Users/Nauman Riaz/Desktop/FYP/test/031_bg-02_144.jpg')
img
learn = load_learner(path)
pred_class, pred_idx, outputs = learn.predict(img)
pred_class

```