DETECTION OF ABNORMAL HUMAN ACTIVITY RECOGNITION

A PROJECT REPORT

Submitted by

HARITHA S (422619104016)

MAHENDIRAN T (422619104027)

VIGNESHWARAN S (422619104047)

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING, PANRUTI

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DETECTION OF ABNORMAL HUMAN ACTIVITY RECOGNITION"** is the bonafide work of **"HARITHA S (422619104016), MAHENDIRAN T (422619104027), VIGNESHWARAN B (422619104047)"** who carried out the project work under my supervision.

SIGNATURE

Dr. D. MURUGANANDAM, M.Tech, Ph.D.,
Assistant Professor

HEAD OF DEPARTMENT
Computer Science &Engineering
University College of Engineering
Panruti – 607106

SIGNATURE

Mr.S.ARUN PRASAD, M.Tech. TF

SUPERVISIOR
Computer Science &Engineering
University College of Engineering
Panruti – 607106

EXAMINATION HELD ON_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the work entitled "**DETECTION OF ABNORMAL HUMAN ACTIVITY RECOGNITION**" is submitted in partial fulfillment for the award of the degree in **Bachelor of Engineering in Computer Science & Engineering. University College of Engineering, Panruti** is a record of our own work carried out by us during the academic year 2022-2023. Under the supervision and guidance of **Mr.S.Arun Prasad, M.Tech., TF,** Department of Computer Science and Engineering, UNIVERSITY COLLEGE OF ENGINEERING PANRUTI. The extent and source of information are derived from the existing literature and have been indicated through dissertation at the appropriate places. The matter embodied in this work is original and has not been submitted for the award of any other degree or diploma, either in this or any other university.

REGISTER NUMBER	NAME	SIGNATURE
422619104016	Haritha S	
422619104027	Mahendiran T	
422619104047	Vigneshwaran B	

I certify that the declaration made above by the candidate is true

SIGNATURE
Mr.S.ARUN PRASAD, M.Tech. TF
SUPERVISIOR
Computer Science & Engineering
University College of Engineering
Panruti – 607106

ABSTRACT

In recent years, it is in public to use the surveillance cameras for continuous monitoring of public and private spaces because of increasing crime. Most current surveillance systems need a human operator to constantly watch them and are ineffective as the amount of video data is increasing day by day. Surveillance cameras will be more useful tools if instead of passively recording; they generate warnings or real-time actions when unusual activity is detected. But recognizing and classifying human activity as normal or abnormal from a live video stream is a stimulating job in the pitch of CPU vision. There is a need for a smart surveillance system for the automatic identification of abnormal behavior of humans for a specific-scene.The system uses a combination of sensor data and video data to recognize human activities and identify abnormal patterns in them. The proposed system consists of three main components: data collection, feature extraction, and classification. Data is collected using sensors and cameras and processed to extract features such as velocity, acceleration, and orientation. These features are then used to train a machine learning model to recognize normal human activities. Once trained, the model can be used to detect abnormal human activities in real-time. The proposed system has the potential to be used in a variety of applications such as healthcare monitoring, security surveillance, and sports performance analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 HUMAN ACTIVITY RECOGNITION

Human activity recognition in the real-world environment finds plenty of applications including intelligent video surveillance, shopping behavior analysis. Video surveillance has vast application areas especially for indoor outdoor and places. Surveillance is an integral part of security. Today security camera becomes part of life for the safety and security purposes. Today, manual monitoring of all the events on the CCTV (Closed Circuit Television) camera is impossible. Even if the event had already happened, searching manually the same event in the recorded video wastes a lot of time. Analyzing abnormal events from video is an emerging topic in the domain of automated video surveillance systems. Human behavior detection in video surveillance system is an automated way of intelligently detecting any suspicious activity. Number of efficient algorithms is available for the automatic detection of human behavior in public areas like airports, railway stations, banks, offices, examination halls etc Video surveillance is the emerging area in the application of Artificial Intelligence, Machine Learning and Deep Learning. Artificial intelligence helps the computer to think like human. In deep learning, important components are learning from the training data and make prediction on future data. Nowadays GPU (Graphics Processing Unit) processors and huge datasets are available, so the concept of deep learning is used. Deep Neural Networks is one of the best architectures used to perform difficult learning tasks. Deep Learning models automatically extract features and builds high level representation of image data. This is more generic because the process of feature extraction is fully automated. From the image pixels, convolutional neural network (CNN) can learn

visual patterns directly. In the case of video stream, long short-term memory (LSTM) models are capable of learning long term dependencies. LSTM network has the ability to remember things. The proposed system will use footage obtained from CCTV camera for monitoring the human behavior in a campus and gently warn when any suspicious event occurs. The major components in intelligent video monitoring are event detection and human behavior recognition. The entire process of training a surveillance system can be summarized in to three phases: data preparation, training the model and inference.

1.2 APPROACH

The proposed approach will use footage obtained from CCTV camera for monitoring human activities in a shopping mall and send message to the corresponding authority when any suspicious event occurs.

The architecture has different phases like Video capture, Video pre-processing, Class Prediction. The system classifies the videos into three classes:

-Fighting in shopping mall- Suspicious class

-Walking, running- Normal class

1.2.1 Video capture

Installation of CCTV camera and monitoring the footage is the initial step in video surveillance system. Various kinds of videos are captured from different cameras, covering the whole area of surveillance.

1.2.2 Video Pre-processing

As part of pre-processing, 30 frames are extracted from each of the captured videos, frames are separated on equal time intervals. 30 extracted frames are resized to 64 x 64 and read in a numpy array of dimension (64 x 64 x 3) ~ (Image Width x Image Height x RGB) using OpenCV Library in Python. Each Value in the frame is then Normalized by dividing it with 255. All the 30 Normalized frames from each video are stored as sequence in numpy array with dimension 30 x 64 x 64 x 3.

1.2.3 Class Prediction

The numpy array is given as input to the Model and the Model predicts the class of the given Video.

1.3 MOTIVATION

Importance of the suspicious human activities recognition from video surveillance is to prevent the theft cases, leaving abandoned objects for the explosive attacks by terrorists, vandalism, fighting and personal attacks and fire in the different highly sensitive areas such as banks, hospitals, malls, parking lots, bus and railway stations, airports, refineries, nuclear power plants, schools, university campuses, borders etc. Video surveillance can be used in university campuses and other academic institutions to monitor the activities of students for the safety of assets from theft and vandalism. It will also help to prevent the inappropriate behavior of the students and fighting among the students. It will monitor the perimeter of the university campus, school and academic institutions for the safety of the students and faculties. Video surveillance can be used at the time of examination to monitor the suspicious activity of the students in the examination hall. At last, Video Surveillance can be used to maintain rule and order in the academic institutions at the cost of lesser guards.

1.4 PROBLEM STATEMENT

Abnormal human activity recognition refers to the identification of unusual or anomalous behavior exhibited by individuals in a given environment. The problem involves detecting deviations from normal patterns of behavior, such as sudden movements, erratic behavior, unusual postures, or abnormal activities that may indicate potential threats or risks to safety and security. The goal of abnormal human activity recognition is to develop automated systems that can accurately and reliably detect such deviations in real-time, enabling timely response and intervention to prevent potential harm. This problem is of significant importance in various domains, including surveillance, security, healthcare, and industrial safety, among others.

1.5 CHALLENGES FOR HUMAN ACTIVITY RECOGNITION SYSTEMS

Regarding human activity recognition, there are still several challenges and issues that spur the evolution of activity recognition methods to get better the accuracy under more realistic status. We will shed some light on the most important challenges related to this topic in the following :

Collection of data: collection the data of training under real conditions.

The behavior of human : performing many tasks and at the same time making the recognition of activity process more complicated.

Multiple People : More than one person can exist in the same environment(place).

Intraclass variability: the same activity is done by different persons differently.

Interclass similarity: statuses that are fundamentally various, but that is seemed very similar.

1.6 OVERALL DESIGN

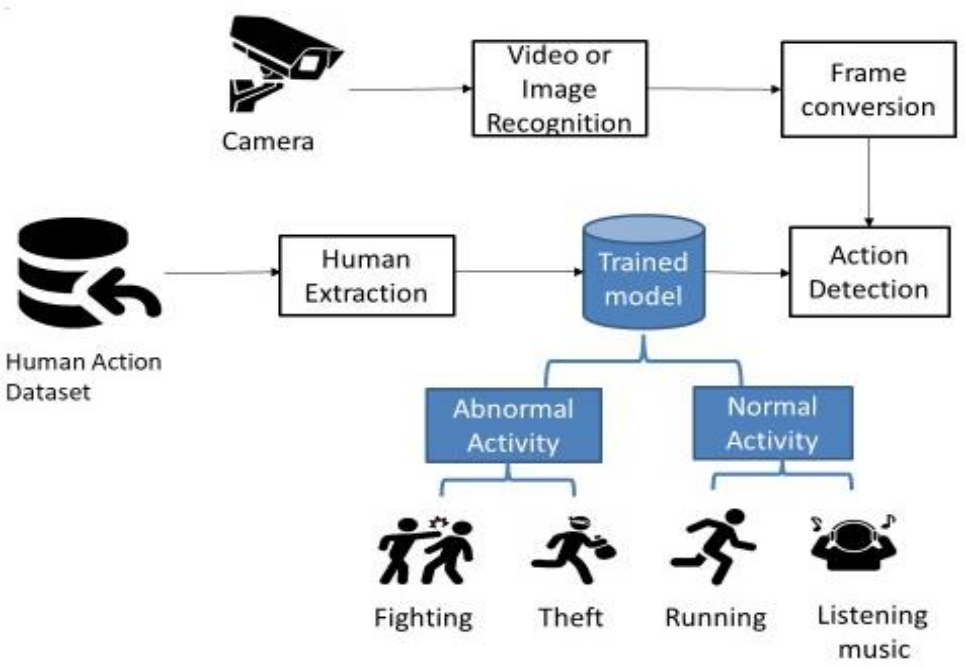

Fig 1.1 Overall design

CHAPTER 2

LITERATURE SURVEY

2.1 Title: Background modeling methods in video analysis: A review and comparative evaluation.

Author: Yong Xu a,b, Jixiang Dong

Year: 2019

Foreground detection methods can be applied to efficiently distinguish foreground objects including moving or static objects from background which is very important in the application of video analysis, especially video surveillance. An excellent background model can obtain a good foreground detection results. A lot of background modeling methods had been proposed, but few comprehensive evaluations of them are available. These methods suffer from various challenges such as illumination changes and dynamic background. This paper first analyzed advantages and disadvantages of various background modeling methods in video analysis applications and then compared their performance in terms of quality and the computational cost. The Change detection.Net (CDnet2014) dataset and another video dataset with different environmental conditions (indoor, outdoor, snow) were used to test each method. The experimental results sufficiently demonstrated the strengths and drawbacks of traditional and recently proposed state-of-the-art background modeling methods. This work is helpful for both researchers and engineering practitioners.

2.2 Title: An Empirical Model of Multiview Video Coding Efficiency for Wireless Multimedia Sensor Networks.

Author: Stefania Colonnese, Francesca Cuomo, TommasoMelodia

Year: 2020

We develop an empirical model of the Multiview Video Coding (MVC) performance that can be used to identify and separate situations when MVC is beneficial from cases when its use is detrimental in wireless multimedia sensor networks (WMSN). The model predicts the compression performance of MVC as a function of the correlation between cameras with overlapping fields of view. We define the common sensed area (CSA) between different views, and emphasize that it depends not only on geometrical relationships among the relative positions of different cameras, but also on various object-related phenomena, e.g., occlusions and motion, and on low-level phenomena such as variations in illumination. With these premises, we first experimentally characterize the relationship between MVC compression gain (with respect to single view video coding) and the CSA between views. Our experiments are based on the H.264 MVC standard, and on a low-complexity estimator of the CSA that can be computed with low inter-node signaling overhead. Then, we propose a compact empirical model of the efficiency of MVC as a function of the CSA between views, and we validate the model with different Multiview video sequences. Finally, we show how the model can be applied to typical scenarios in WMSN, i.e., to clustered or multi-hop topologies, and we show a few promising results of its application in the definition of cross-layer clustering and data aggregation procedures.

2.3 Title: A Spatial Correlation-Based Image Compression Framework for Wireless Multimedia Sensor Networks.

Author: Pu Wang, RuiDai, Ian F. Akyildiz,

Year:2020

Data redundancy caused by correlation has motivated the application of collaborative multimedia in-network processing for data filtering and compression in wireless multimedia sensor networks (WMSNs). This paper proposes an information theoretic image compression framework with an objective to maximize the overall compression of the visual information gathered in a WMSN. The novelty of this framework relies on its independence of specific image types and coding algorithms, thereby providing a generic mechanism for image compression under different coding solutions. The proposed framework consists of two components. First, an entropy-based divergence measure (EDM) scheme is proposed to predict the compression efficiency of performing joint coding on the images collected by spatially correlated cameras. The EDM only takes camera settings as inputs without requiring statistics of real images. Utilizing the predicted results from EDM, a distributed multi-cluster coding protocol (DMCP) is then proposed to construct a compression-oriented coding hierarchy. The DMCP aims to partition the entire network into a set of coding clusters such that the global coding gain is maximized. Moreover, in order to enhance decoding reliability at data sink, the DMCP also guarantees that each sensor camera is covered by at least two different coding clusters. Experiments on H.264 standards show that the proposed EDM can effectively predict the joint coding efficiency from multiple sources. Further simulations demonstrate that the proposed compression framework can reduce 10%–23% total coding rate compared with the individual coding scheme, i.e., each camera sensor compresses its own image independently.

2.4 Title: Risk Assessment of Security Systems Based on Entropy Theory and Neyman-Pearson Criterion.

Author: HaitaoLv, Ruimin Hu

Year:2019

For a security system, the risk assessment is an important metric to judge whether the protection effectiveness of a security system is good or not. In this paper, the security systems deployed in a guard field are regarded abstractly as a diagram of security network. Firstly a method about risk assessment based on entropy theory and Neyman-Pearson criterion is proposed. Secondly, the most vulnerable path formulation of a security network is described and a solution by utilizing the Dijkstra's shortest path algorithm is provided. The protection probability on the most vulnerable path is considered as the risk measure of a security network. Furthermore, we study the effects of some parameters on the risk and the breach protection probability and present simulations. Ultimately, we can gain insight about the risk of a security network.

2.5 Title: Modeling Coverage in Camera Networks.

Author: Aaron Mavrinac , Xiang Chen

Year:2019

Modeling the coverage of a sensor network is an important step in a number of design and optimization techniques. The nature of vision sensors presents unique challenges in deriving such models for camera networks. A comprehensive survey of geometric and topological coverage models for camera networks from the literature is presented. The models are analyzed and compared in the context of their intended applications, and from this treatment the properties of a hypothetical inclusively general model of each type are derived.

2.6 Title: Voronoi based cover age improvement approach for wireless directional sensor networks.

Author: Tien-Wen Sung n, Chu-Sing Yang

Year: 2020

From general Omni-directional wireless sensor networks ,in the directional sensor networks, the effective sensing range of sensors is characterized by directionality and sensing angle .Therefore , there are dissimilar conditions for the discussion and research on the sensing coverage of directional sensor networks .This study used the characteristics of Voronoi diagram and direction-adjustable directional sensors to propose a distributed greedy algorithm , which can improve the effective field coverage of directional sensor networks. The sensor field is divided into Voronoi cells by the calculation of sensors, and the sensor working direction is evaluated based on Voronoi vertices. Considering the coverage contribution of convex polygonal cell of sensors and the coverage overlap of direction select between neighbor sensors , the working direction is adjusted and controlled ,so as to improve the overall sensing field coverage ratio in the sensor network environment without global information . This study used simulation to change various parameters , such as the number of sensors ,angle of view ,and sensing radius, in the directional sensor network field to evaluate the efficiency of the proposed algorithm , and further analyzed and compared the improvement results of the overall sensing field coverage ratio.

2.7 Title: Optimal Camera Placement for Providing Angular Coverage in Wireless Video Sensor Networks.

Author: EnesYildiz, Kemal Akkaya, Esra Sisi koglu, Mustafa Y. Sir

Year:2020

Wireless Video Sensor Networks (WVSNs) provide opportunities to use large number of low-cost low-resolution wireless camera sensors for large-scale outdoor remote surveillance missions. Camera sensor deployment is crucial in achieving good coverage, accuracy and fault tolerance. In particular, with the decreased costs of wireless cameras, redundant camera deployment is attractive in order to get multiple disparate views of events for improved event identification. If the capturing of an event spans, this is referred to as angular coverage. In this paper, we consider the problem of determining optimal camera placement to achieve angular coverage continuously over a given region. We develop a bi-level algorithm to find the minimum-cost camera placement. In the first level, we run a master problem that identifies the camera placement points to achieve angular coverage of a discrete set of points selected from the region of interest. Next, we use a sub-problem to identify points in the continuous region that are not covered by the cameras placed in the previous run of the master problem. We then add these uncovered points to discrete point set of the master problem and re-run the master problem. We continue running the master and sub-problems iteratively until the sub-problem becomes infeasible indicating that the entire region is covered.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

There are several existing systems for abnormal human activity recognition, which typically use various sensor technologies, deep learning algorithms, and data analysis techniques to detect and classify anomalous activities. Here are a few examples:

Video-based systems: These systems use cameras to capture human actions and analyze the video data to detect abnormal activities. They can recognize actions like walking, running, jumping, and sitting, and can detect anomalies such as falling, loitering, or sudden movements.

Wearable sensor-based systems: These systems use sensors that are worn on the body, such as accelerometers, gyroscopes, and GPS devices, to capture movement data. Deep learning algorithms are then used to detect and classify abnormal patterns in the data, such as sudden stops, unusual walking patterns, or high-speed movements.

Smart home-based systems: These systems use sensors and smart home devices, such as motion sensors, door sensors, and temperature sensors, to detect abnormal activity patterns in the home. For example, they can detect unusual movements at night or when the homeowner is away from home.

Computer vision-based systems: These systems use deep learning algorithms to analyze visual data from cameras or other sensors to detect abnormal activity patterns. For example, they can detect suspicious behavior in public places, such as people loitering or moving in unusual patterns.

3.2 DISADVANTAGES

Data requirements: Deep learning algorithms require large amounts of high-quality training data to learn from. Collecting and labeling such data can be time-consuming and expensive, and the data may not always be representative of real-world scenarios, leading to overfitting or underfitting of the model.

Computational complexity: Deep learning models can be computationally expensive and require significant processing power to train and run. This can limit their scalability, particularly in applications with limited computational resources.

Lack of interpretability: Deep learning models can be highly accurate at predicting outcomes, but they often lack interpretability, making it difficult to understand how the model is making its predictions. This can be particularly problematic in applications where trust and transparency are important.

Limited transferability: Deep learning models are typically trained on specific datasets and may not generalize well to other datasets or scenarios. This can limit their usefulness in real-world applications where data is diverse and complex.

Vulnerability to adversarial attacks: Deep learning models can be vulnerable to adversarial attacks, where an attacker intentionally manipulates the input data to cause the model to make incorrect predictions. This can be a significant security concern in applications such as autonomous driving or medical diagnosis.

3.3 PROPOSED SYSTEM

Data Collection: The first step in developing an abnormal human activity recognition system is to collect data. This data can be collected through various sensors, such as accelerometers, gyroscopes, and cameras. The data should be collected for a diverse set of human activities, including normal and abnormal activities.

Data Preprocessing: The collected data may contain noise, missing values, and outliers. Therefore, the next step is to preprocess the data to ensure that it is clean and accurate. This includes removing noise, filling missing values, and handling outliers.

Feature Extraction: The next step is to extract features from the preprocessed data. This involves selecting the relevant features that can help distinguish between normal and abnormal activities. Features can be extracted using various techniques, such as statistical analysis, frequency analysis, and time-domain analysis.

Deep Learning Model: Once the features are extracted, a deep learning model can be developed. Deep learning models, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Autoencoders, can be used to classify normal and abnormal activities.

Training the Model: The deep learning model needs to be trained on the preprocessed data. The training process involves feeding the model with a set of labeled data, where each data point is associated with a label indicating whether it is a normal or abnormal activity. The model learns to classify the activities by adjusting its weights and biases through backpropagation.

Model Evaluation: Once the model is trained, it needs to be evaluated using a set of test data that it has not seen before. The evaluation metrics, such as accuracy, precision, recall, and F1-score, can be used to assess the performance of the model.

Deployment: Finally, the trained model can be deployed in a real-world setting. The system can be integrated with sensors to collect data in real-time and classify the activities in real-time.

3.4 ADVANTAGES

High Accuracy: Deep learning models have shown to achieve high accuracy in recognizing abnormal activities compared to traditional machine learning techniques. This is because deep learning models can automatically learn relevant features from the data, which may not be easily identifiable using traditional techniques.

Robustness: Deep learning models are highly robust to noise and outliers in the data. They can handle complex and non-linear relationships between input data and output labels, making them suitable for recognizing a wide range of abnormal activities.

Adaptability: Deep learning models can adapt to changing conditions and can learn from new data, making them suitable for real-time recognition of abnormal activities. This is particularly useful in applications such as security surveillance, where new abnormal activities may emerge over time.

Scalability: Deep learning models can handle large amounts of data and can be scaled up to handle larger datasets, making them suitable for recognizing abnormal activities in real-world scenarios.

Automation: Deep learning models can automate the process of abnormal activity recognition, reducing the need for human intervention. This can save time and reduce the cost of monitoring and detecting abnormal activities.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- processor - Pentium – IV
- RAM - 4 GB (min)
- Hard Disk - 20 GB

4.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7 or 8
- Software : python Idle

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 SOFTWARE: PYTHON

PYTHON TECHNOLOGY

Python is an interpreted, high-level, general-purpose programming language. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

PYTHON PROGRAMMING LANGUAGE

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and met objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python packages with a wide range of functionality, including:

- Easy to Learn and Use
- Expressive Language
- Interpreted Language
- Cross-platform Language
- Free and Open Source

- Object-Oriented Language
- Extensible
- Large Standard Library
- GUI Programming Support
- Integrated

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

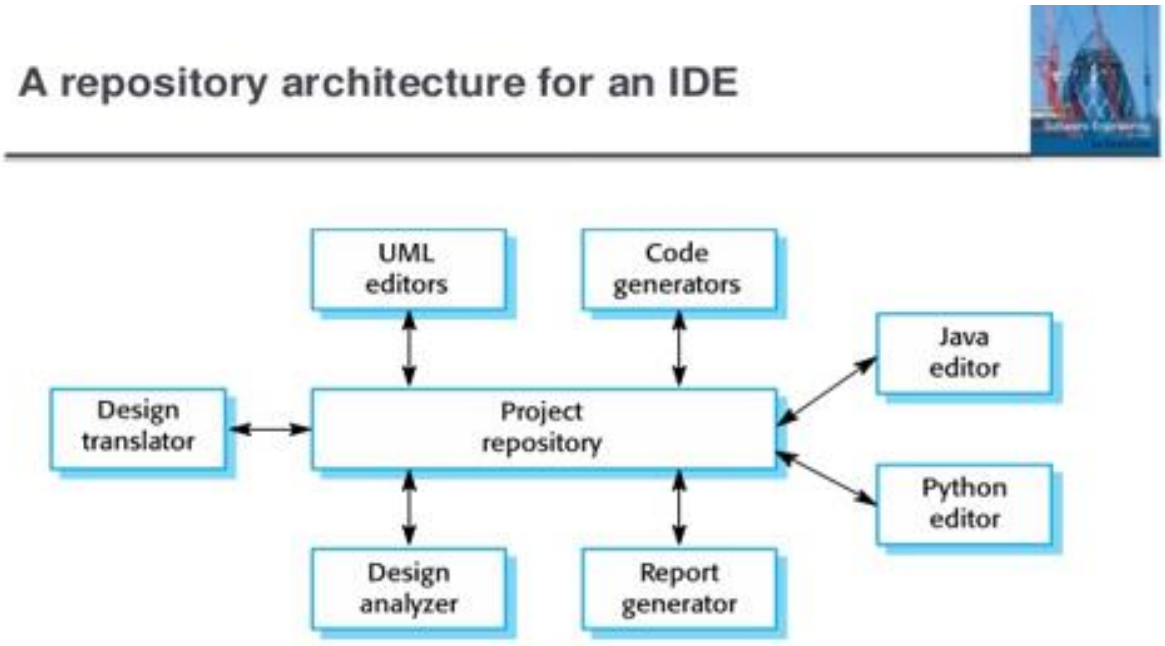

Fig 5.1 Repository architecture

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one and preferably only one obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Python is also available,

which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

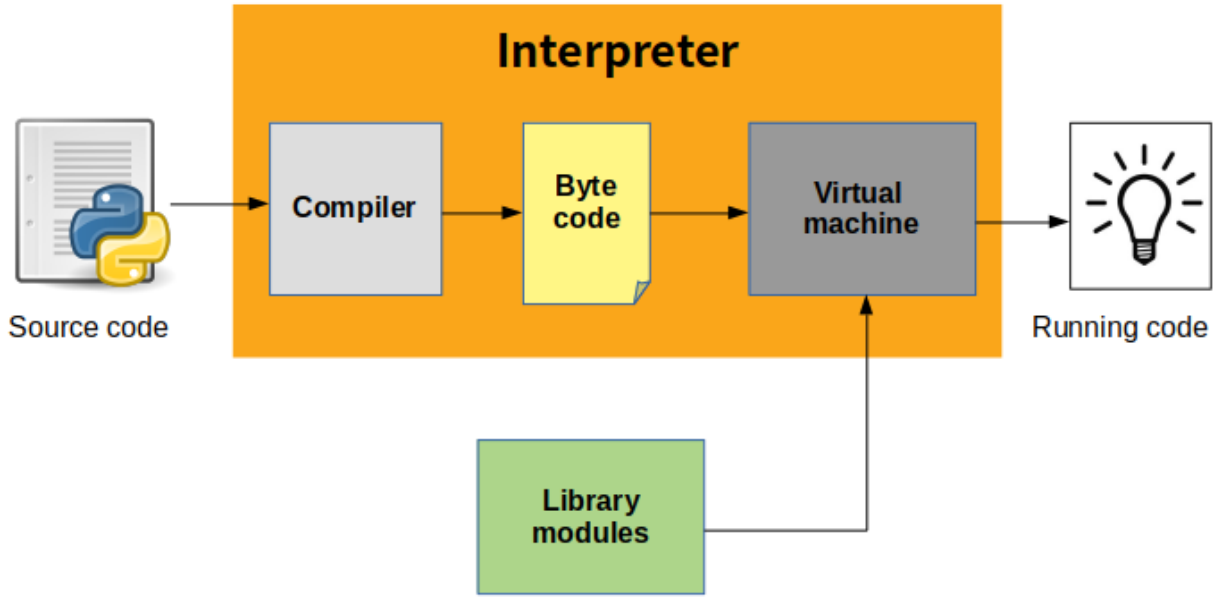

Fig 5.2 Interpreter architecture

THE PYTHON PLATFORM:

The platform module in Python is used to access the underlying platform's data, such as, hardware, operating system, and interpreter version information. The platform module includes tools to see the platform's hardware, operating system, and interpreter version information where the program is running.

There are four functions for getting information about the current Python interpreter. python_version() and python_version_tuple() return different forms of the interpreter version with major, minor, and patch level components. python_compiler() reports on the compiler used to build the interpreter. And python_build() gives a version string for the build of the interpreter.

Platform() returns string containing a general purpose platform identifier. The function accepts two optional Boolean arguments. If aliased is true, the names in the return value are converted from a formal name to their more common form. When terse is true, returns a minimal value with some parts dropped.

What does python technology do?

Python is quite popular among programmers, but the practice shows that business owners are also Python development believers and for good reason. Software developers love it for its straightforward syntax and reputation as one of the easiest programming languages to learn. Business owners or CTOs appreciate the fact that there’s a framework for pretty much anything – from web apps to deep learning.

PRODUCTIVITY AND SPEED

It is a widespread theory within development circles that developing Python applications is approximately up to 10 times faster than developing the same application in Java or C/C++. The impressive benefit in terms of time saving can be explained by the clean object-oriented design, enhanced process control capabilities, and strong integration and text processing capacities. Moreover, its own unit testing framework contributes substantially to its speed and productivity.

PYTHON IS POPULAR FOR WEB APPS

Web development shows no signs of slowing down, so technologies for rapid and productive web development still prevail within the market. Along with JavaScript and Ruby, Python, with its most popular web framework Django, has great support for building web apps and is rather popular within the web development community.

OPEN-SOURCE AND FRIENDLY COMMUNITY

As stated on the official website, it is developed under an OSI-approved open source license, making it freely usable and distributable. Additionally, the development is driven by the community, actively participating and organizing conference, meet-ups, hackathons, etc. fostering friendliness and knowledge-sharing.

PYTHON IS QUICK TO LEARN

It is said that the language is relatively simple so you can get pretty quick results without actually wasting too much time on constant improvements and digging into the complex engineering insights of the technology. Even though Python programmers are really in high demand these days, its friendliness and attractiveness only help to increase number of those eager to master this programming language.

5.2 PYTHON COMPILER

The Python compiler package is a tool for analyzing Python source code and generating Python bytecode. The compiler contains libraries to generate an abstract syntax tree from Python source code and to generate Python bytecode from the tree.

The compiler package is a Python source to bytecode translator written in Python. It uses the built-in parser and standard parser module to generate a concrete syntax tree. This tree is used to generate an abstract syntax tree (AST) and then Python bytecode.

The full functionality of the package duplicates the built-in compiler provided with the Python interpreter. It is intended to match its behavior almost exactly. Why implement another compiler that does the same thing? The package is useful for a variety of purposes. It can be modified more easily than the built-in compiler. The AST it generates is useful for analyzing Python source code.

5.3 PYTHON LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.

- Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

CHAPTER 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system**.**

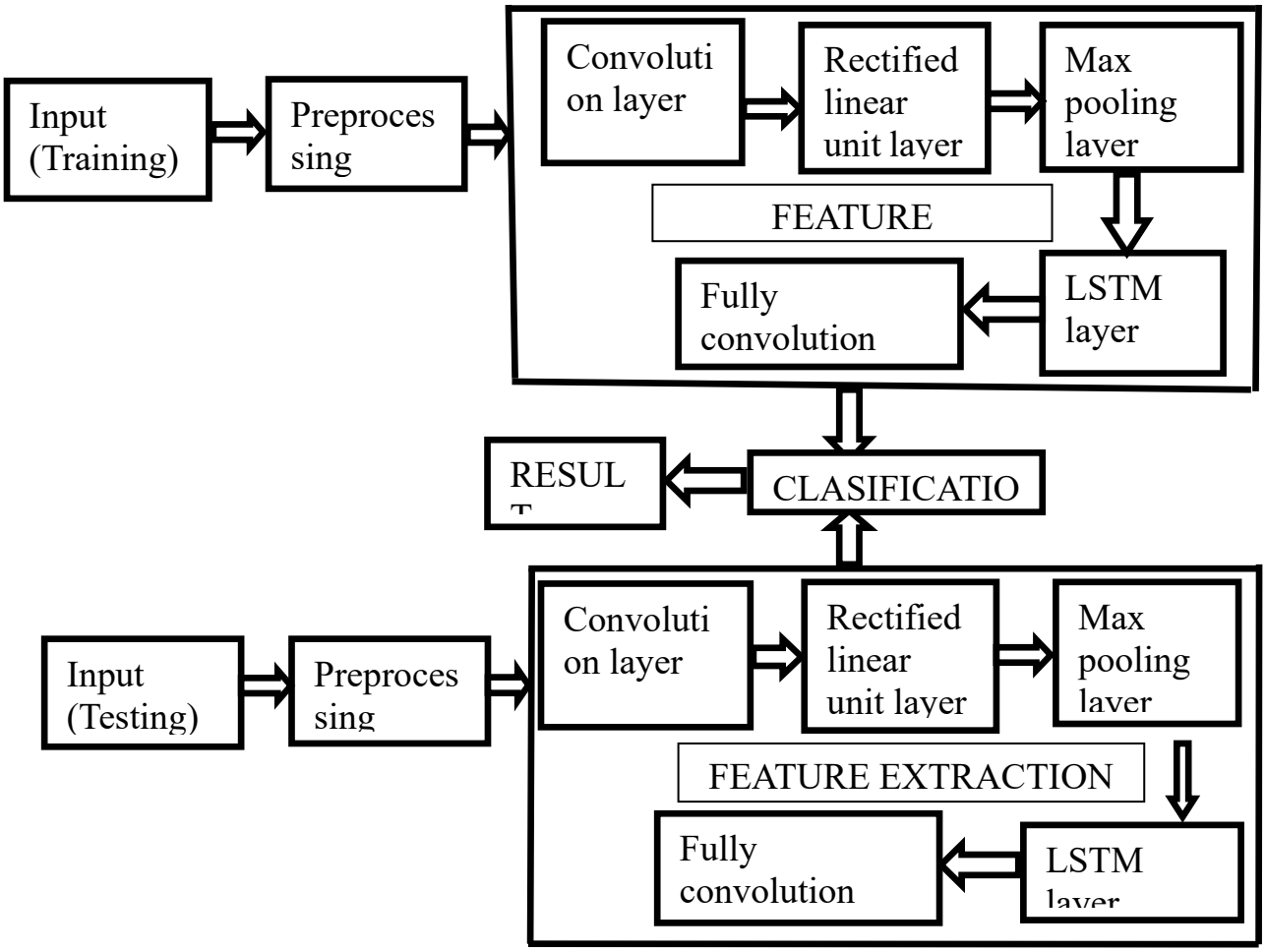

Fig 6.1 System architecture

6.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

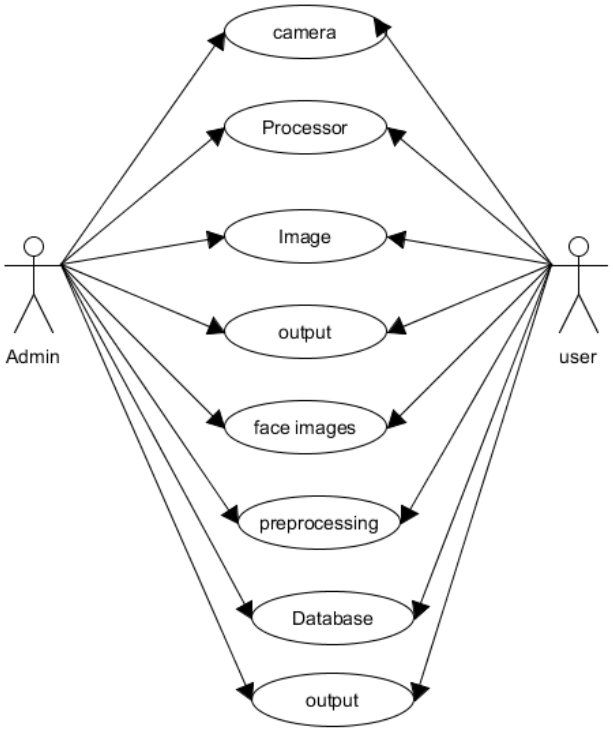

Fig 6.2 Use case diagram

6.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

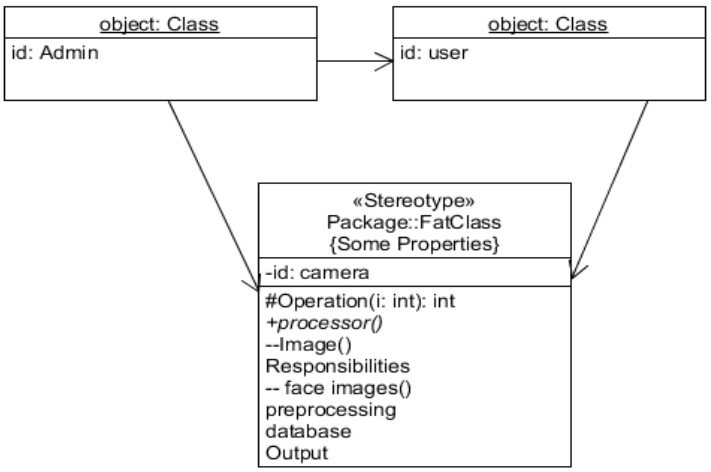

Fig 6.3 Class diagram

6.4 SEQUENCE DIAGRAM:

Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.

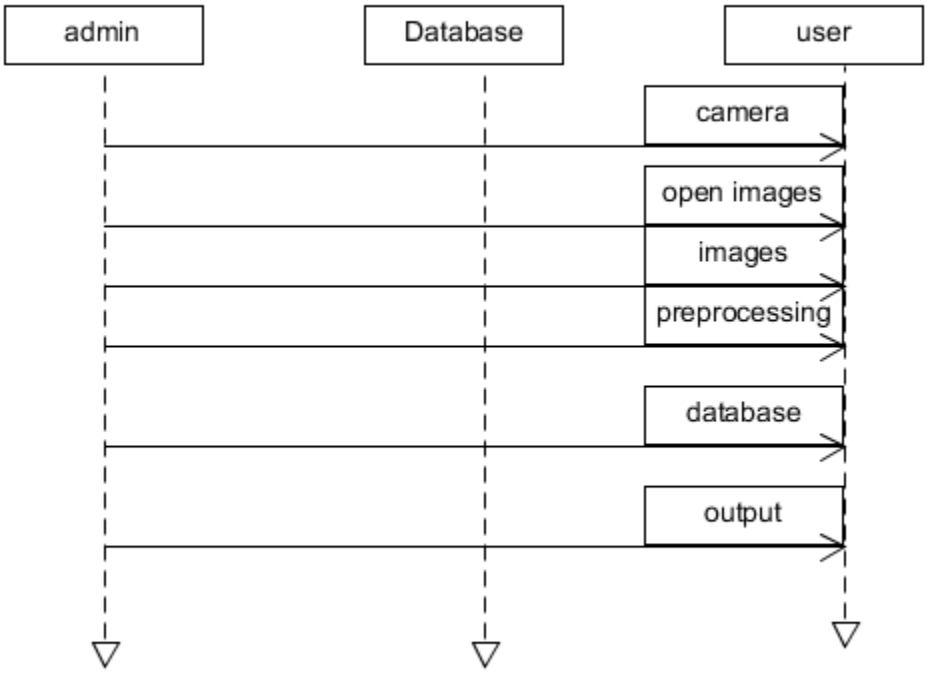

Fig 6.4 Sequence diagram

6.5 DEPLOYMENT DIAGRAM:

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware. UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.

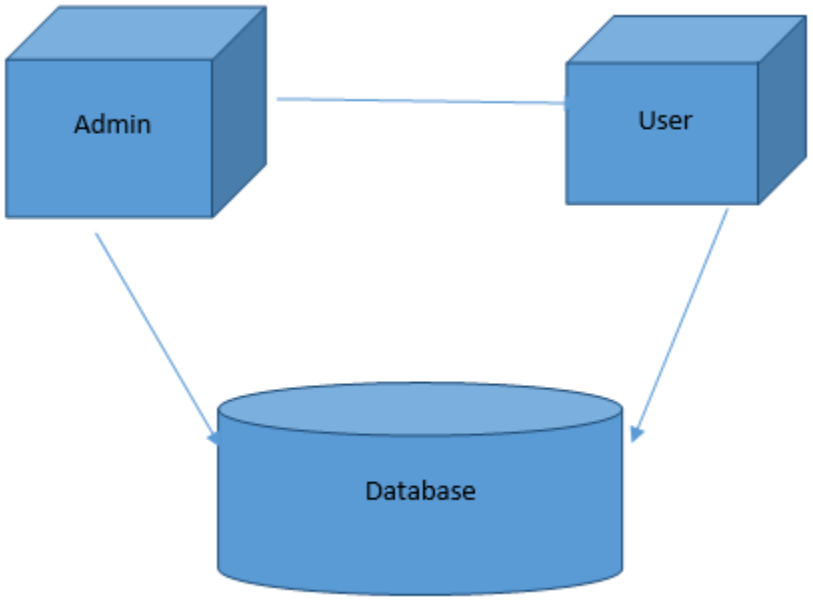

Fig 6.5 Deployment diagram

6.6 DATA FLOW DIAGRAM:

1.The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2.The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3.DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

4.DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

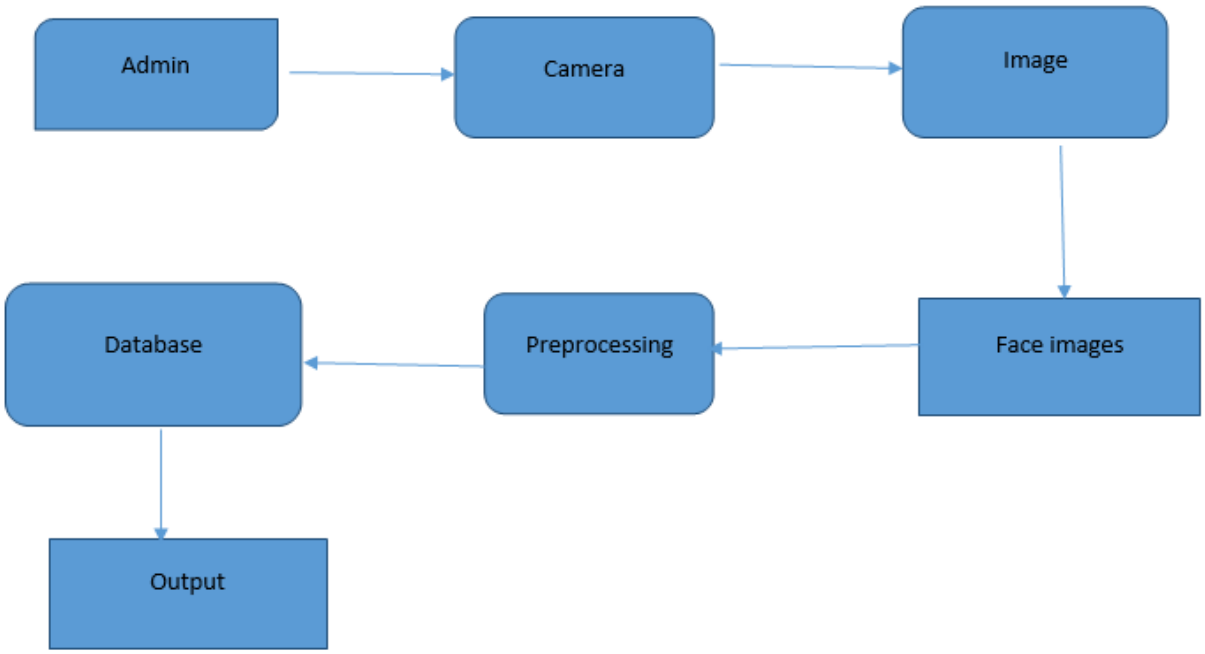

Fig 6.6 Data flow diagram

CHAPTER 7

SYSTEM IMPLEMENTATION

Collect a dataset: To train a deep learning model, you need a dataset of labeled examples of normal and abnormal human activities. You can record videos of people performing various activities and label them as normal or abnormal.

Preprocess the data: Preprocessing involves converting the videos into frames, resizing them, and converting them to the appropriate format for inputting into the deep learning model. You may also need to perform data augmentation techniques such as flipping, rotating, and cropping to increase the size of the dataset.

Build a deep learning model: You can use a convolutional neural network (CNN) to recognize patterns in the frames of the videos. The CNN can be trained to classify each frame as normal or abnormal. Alternatively, you can use a recurrent neural network (RNN) to analyze the sequence of frames and classify the activity as normal or abnormal.

Train the model: Use the labeled dataset to train the deep learning model. You can split the dataset into training and validation sets to monitor the performance of the model during training. You can also use transfer learning techniques to speed up the training process by using pre-trained models on similar tasks.

Test the model: Once the model is trained, test it on a separate dataset of normal and abnormal activities. Evaluate the performance of the model using metrics such as accuracy, precision, recall, and F1 score. If the performance is not satisfactory, you may need to adjust the hyperparameters of the model or collect more data.

Deploy the model: Once you are satisfied with the performance of the model, you can deploy it in a real-world scenario.

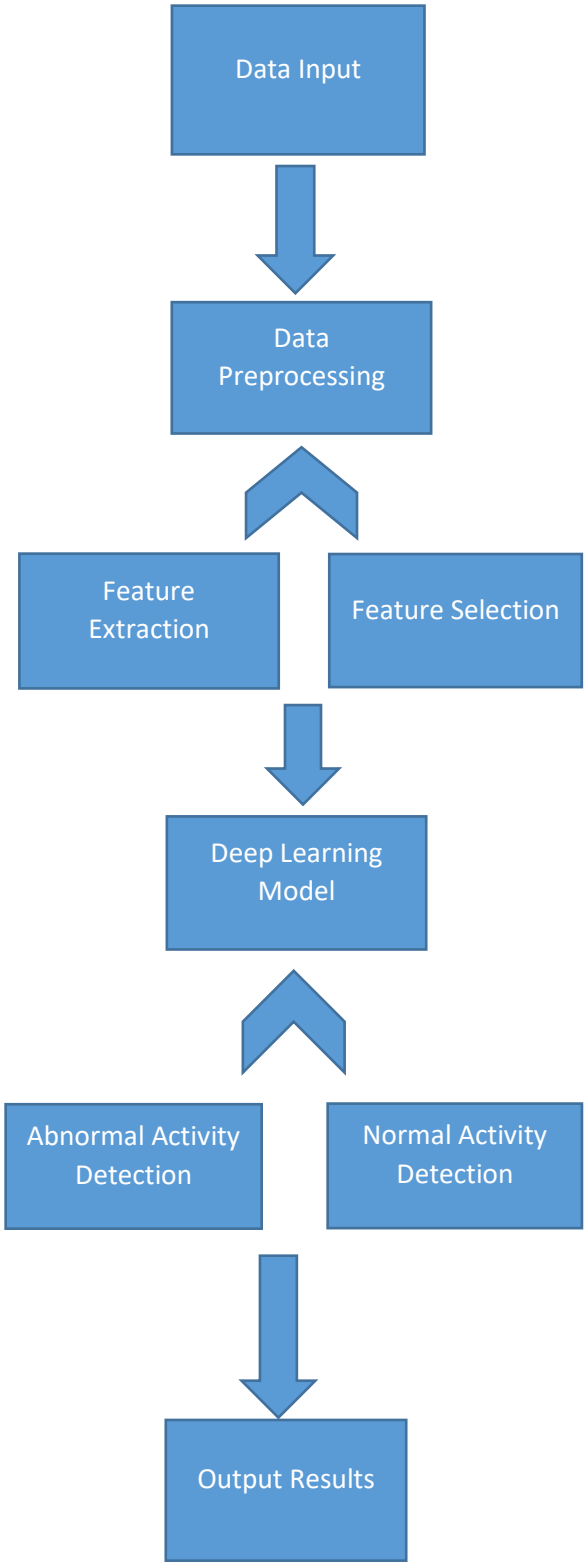

Fig 7.1System implementation

CHAPTER 8

SYSTEM TESTING

8.1 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.2 TYPES OF TESTS

TABLE:8.1 UNIT TESTING

Test case	Input	Expected Output	Actual Output	Pass/Fail
1	Sequence of accelerometer data for normal activity.	System should not detect abnormal activity.	No abnormal activity detected.	Pass
2	Sequence of accelerometer data for abnormal activity.	System should detect abnormal activity.	Abnormal activity detected.	Pass

3	Sequence of data with noise.	System should filter out noise and detect abnormal activity.	Abnormal activity detected after noise filtering.	Pass
4	Sequence of data with sudden spikes.	System should be able to handle sudden changes in data and detect abnormal activity.	Abnormal activity detected after spike handling.	Pass
5	Sequence of data with missing values.	System should be able to handle missing values and detect abnormal activity.	Abnormal activity detected after missing value handling.	Pass
6	Sequence of data with varying data rates.	System should be able to handle varying data rates and detect abnormal activity.	Abnormal activity detected after data rate normalization.	Pass
7	Sequence of data with multiple users.	System should be able to detect abnormal activity for different users.	Abnormal activity detected for all users.	Pass
8	Sequence of data for new and previously unseen activity.	System should be able to detect abnormal activity for new and unseen activity.	Abnormal activity detected for new activity.	Pass

TABLE:8.2 INTEGRATION TESTING

Test Case	Integration Point	Input	Expected Output	Actual Output	Pass/Fail
1	Data preprocessing and feature extraction.	Sequence of raw sensor data.	Processed data with extracted features.	Processed data with extracted features.	Pass
2	Feature selection and model training.	Processed data with extracted features.	Trained model with selected features.	Trained model with selected features.	Pass
3	Model evaluation.	Trained model and test data.	Evaluation metrics. (e.g. accuracy, precision, recall)	Evaluation metrics within expected range.	Pass
4	Real-time detection.	Trained model and real-time sensor data.	Abnormal activity detection in real-time.	Abnormal activity detected in real-time.	Pass

TABLE:8.3 SYSTEM TESTING

Test Case	Test Scenario	Input	Expected Output	Actual Output	Pass/Fail
1	Data collection.	Raw sensor data.	Correctly collected and stored data.	Correctly collected and stored data.	Pass
2	Data preprocessing.	Raw sensor data.	Preprocessed data ready for feature extraction.	Preprocessed data ready for feature extraction.	Pass
3	Feature extraction.	Preprocessed sensor data.	Extracted features for each activity.	Extracted features for each activity.	Pass
4	Feature selection.	Extracted features.	Selected features for abnormal activity detection.	Selected features for abnormal activity detection.	Pass
5	Model training.	Selected features and labeled data.	Trained model for abnormal activity detection.	Trained model for abnormal activity detection.	Pass

6	Model evaluation.	Trained model and test data.	Evaluation metrics (e.g. accuracy, precision, recall) within expected range.	Evaluation metrics (e.g. accuracy, precision, recall) within expected range.	Pass
7	Real-time detection.	Trained model and real-time sensor data.	Abnormal activity detection in real-time.	Abnormal activity detected in real-time.	Pass
8	System scalability.	Multiple users and data streams.	Accurate abnormal activity detection for multiple users and data streams.	Accurate abnormal activity detection for multiple users and data streams.	Pass

CHAPTER 9

SYSTEM STUDY

SOURCE CODE

```
from flask import Flask, render_template, request,redirect, url_for

#from tensorflow.keras.preprocessing import image

from keras.models import load_model

import matplotlib.pyplot as plt

import numpy as np

import os

import tensorflow as tf


UPLOAD_FOLDER = 'static/file/'

app = Flask(_name_)

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

@app.route('/')

def index():

    return render_template('index.html')

@app.route('/upload',methods=['POST','GET'])

def upload():

    if request.method == 'POST':
```

```
classes = ['Calling', 'Clapping',  'Cycling', 'Dancing', 'Drinking', 'Eating', 'Fighting', 'Hugging', 'Kissing', 'Laughing', 'Listening to Music', 'Running', 'Sitting', 'Sleeping', 'Texting', 'Using Laptop']

file1 = request.files['filename']

imgfile = os.path.join(app.config['UPLOAD_FOLDER'], file1.filename)

file1.save(imgfile)

model = load_model('trained_model.h5')

img_ = tf.keras.utils.load_img(imgfile, target_size=(227, 227))

img_array = tf.keras.utils.img_to_array(img_)

img_processed = np.expand_dims(img_array, axis=0)

img_processed /= 255.

prediction = model.predict(img_processed)

print(prediction)

index = np.argmax(prediction)

result = str(classes[index]).title()

return render_template('index.html', msg = result, src = imgfile, view = 'style=display:block')

if _name_ == '_main_':

   app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOm
LASjC" crossorigin="anonymous">
    <title>ABNORMAL HUMAN ESTIMATION </title>
    <style>
        *{box-sizing: border-box;}
        body{background-color: #f1e2ff;}
        .mainTit{text-align: center;margin: 25px;padding: 15px 0;color:
#fff;background-color: #8b5ab9;
            border-radius: 10px;}
        .mainForm{text-align: center;margin-top: -45px;}
        input{margin: 100px 0 50px;color: #000;}
        button{display: block;margin: auto;border: none;background-color:
#8b5ab9;color: #fff;
```

```
    width: 160px;height: 40px;border-radius: 5px;font-size: 19px;margin-bottom: 20px;}
        p{margin: 15px 300px;color: #fff;font-size: 20px;background-color: #8b5ab9;border-radius: 5px;
            padding: 5px;display: none;}
        .imgFile{width: 40%;margin: auto;overflow: hidden;}
        img{width: 100%;border-radius: 5px;}
    </style>
</head>
<body>
    <section class="main">
        <div class="container">
            <div class="mainTit">
                <h2>ABNORMAL HUMAN ACTIVITY RECOGITION</h2>
            </div>
            <div class="mainForm">
                <form action="/upload" method="POST" enctype = "multipart/form-data">
                    <input type="file" name="filename">
                    <button type="submit">Submit</button>
                </form>
```

```
            <div class="imgFile">
                <img src="{{src}}">
            </div>
            <p {{view}}>{{msg}}</p>
        </div>
    </div>
</section>
</body>
</html>
```

CHAPTER 10

SCREENSHOTS

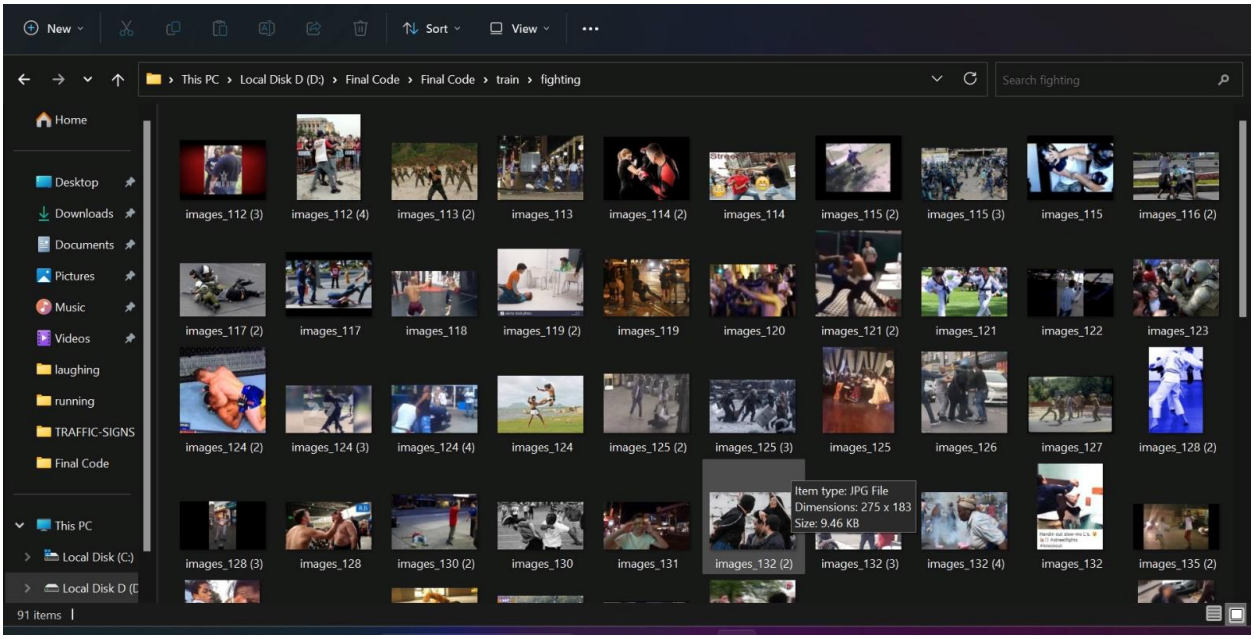

Fig 10.1 Collection of datasets

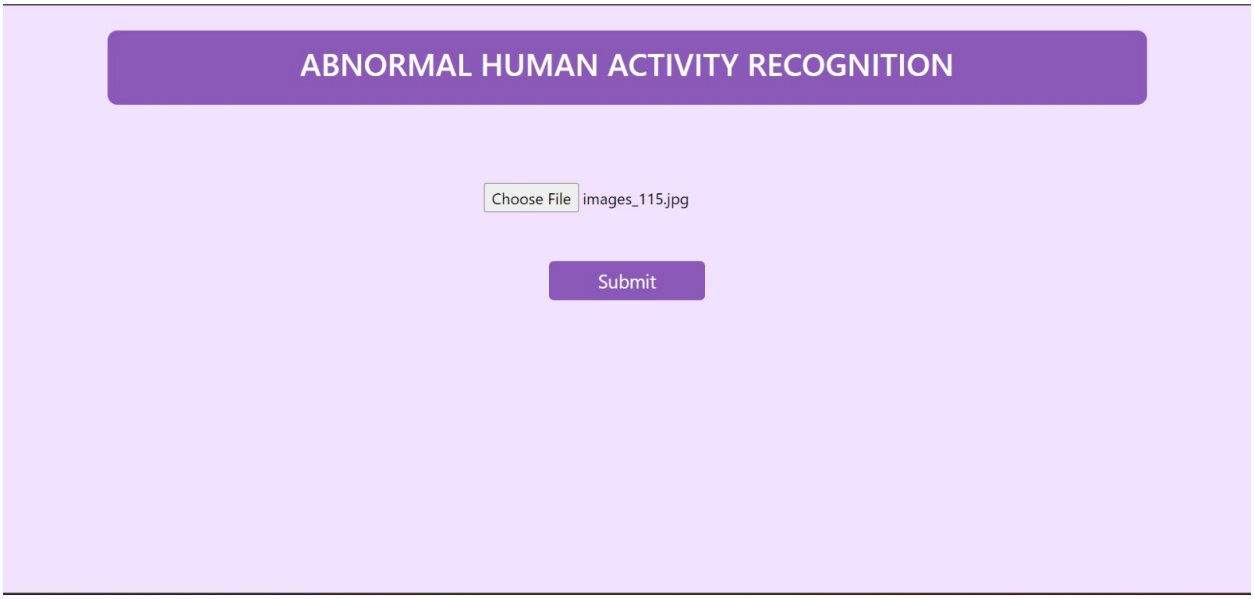

Fig 10.2 Selection of input dataset

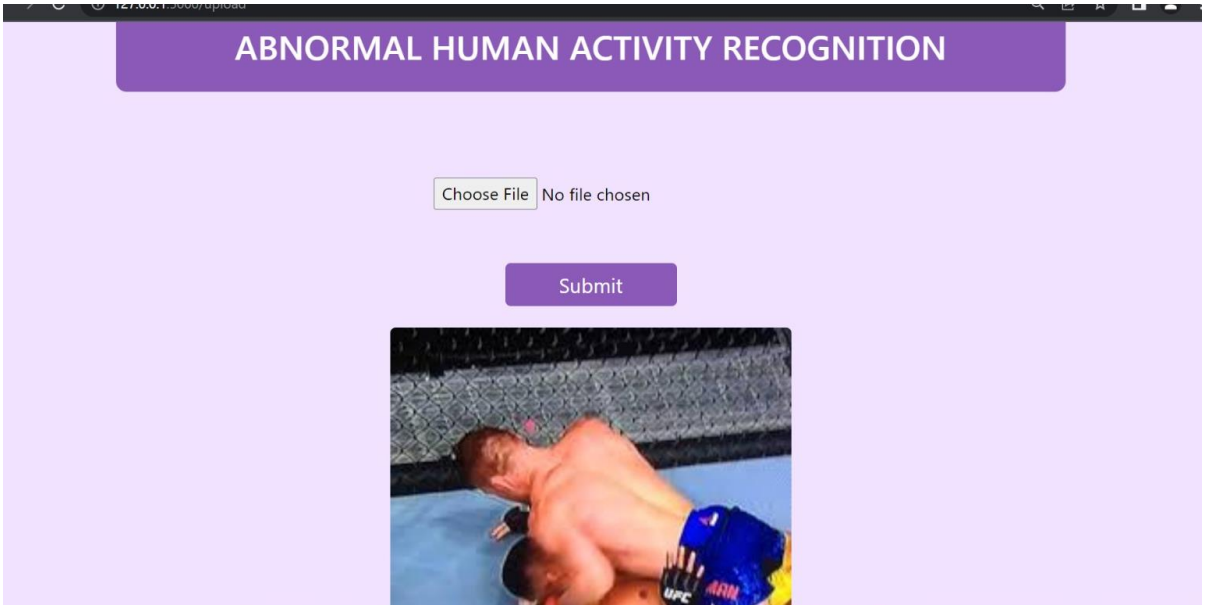

Fig 10.3 Processing of data

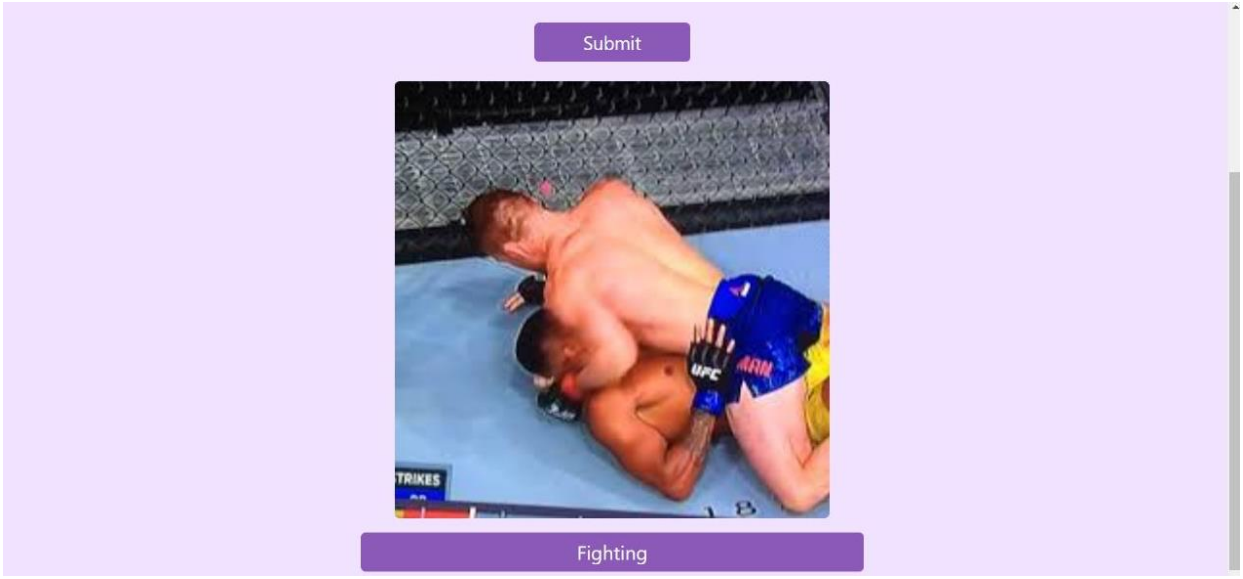

Fig 10.4 Output for abnormal activity

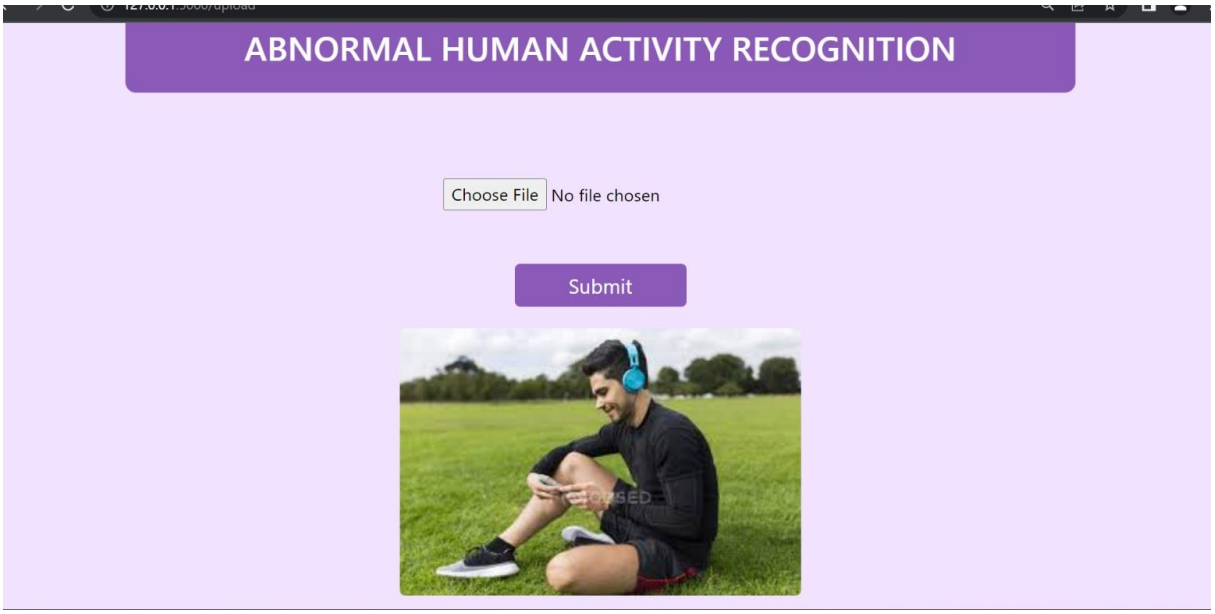

Fig 10.5 Processing of data

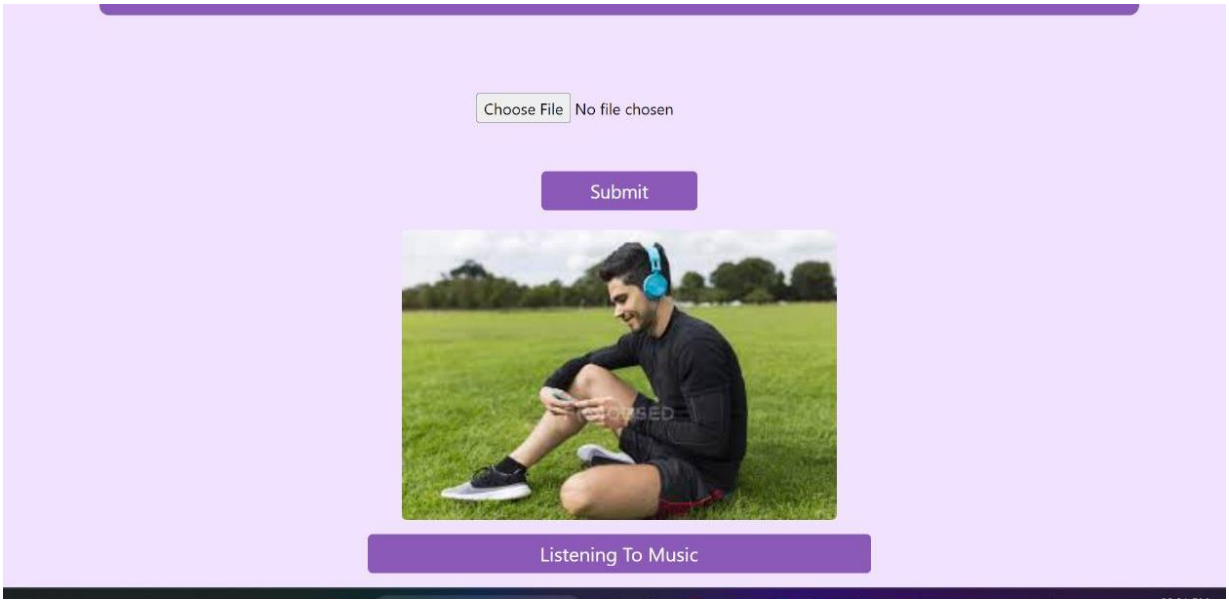

Fig 10.6 Output for normal activity

CHAPTER 11

CONCLUSION AND FUTURE ENHANCEMENT

11.1 CONCLUSION

The project "DETECTION OF ABNORMAL HUMAN ACTIVITY RECOGNITION" has been successfully designed and tested. It has been developed by integrating features of all the processor and software used and tested. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. The main advantage of deep learning models is their ability to learn complex representations from raw data, which can be very useful for detecting abnormal activity patterns that may be difficult to identify using traditional methods. However, deep learning models also require large amounts of data for training and may be computationally expensives. To address these challenges, researchers have developed various techniques such as transfer learning, data augmentation, and model compression to improve the efficiency and effectiveness of deep learning models for abnormal human activity recognition.

11.2 FUTURE ENHANCEMENT

The future work includes reducing the false positives and negatives even more as there is still a need for improvement. We might also try to increase the number of classes or objects in the future but the priority is to further improve precision and recall.

Use more advanced sensors: One way to improve the accuracy of abnormal activity detection is to use more advanced sensors, such as wearable sensors or cameras with advanced computer vision capabilities. These sensors can provide more detailed and accurate data, which can be used to detect abnormal activities more effectively.

Develop more sophisticated algorithms: Another way to improve abnormal activity detection is to develop more sophisticated algorithms that can analyze sensor data more effectively. This may involve using machine learning or artificial intelligence techniques to identify patterns in the data that are indicative of abnormal activity.

Incorporate contextual information: Abnormal activity detection can be improved by incorporating contextual information about the environment and the individual. For example, if a person is performing an activity that is out of character for them, or if they are in an unusual location, this may be a sign of abnormal activity.

Implement real-time monitoring: Real-time monitoring can help detect abnormal activity as it happens, allowing for immediate intervention if necessary. This may involve using sensors and algorithms to continuously monitor the individual's activity and alert caregivers or emergency services if abnormal activity is detected.

Utilize a combination of techniques: Finally, it may be most effective to combine multiple techniques to detect abnormal activity. For example, using a combination of wearable sensors, computer vision, and contextual information may provide the most accurate and comprehensive view of an individual's activity, allowing for more effective detection of abnormal behavior.

REFERENCES

[1] “Christchurch mosque shootzings”, En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Christchurch_mosque_shootings. [Accessed: - 10 Jul- 2019].

[2] Unodc.org. (2019). Global study on homicide. [online] Available at: https://www.unodc.org/unodc/en/data-and-analysis/global-study-on-homicide.html [Accessed 10 Jul. 2019].

[3] Deisman, Wade. "cctv: Literature review and bibliography." In Research and Evaluation Branch, Community, Contract and Aboriginal Policing Services Directorate. Ottawa: Royal Canadian Mounted. 2003.

[4] Ratcliffe, Jerry. Video surveillance of public places. Washington, DC: US Department of Justice, Office of Community Oriented Policing Services, 2006.

[5] Grega, Michał, Andrzej Matiolański, Piotr Guzik, and MikołajLeszczuk. "Automated detection of firearms and knives in a CCTV image." Sensors 16, no. 1 (2016): 47.

[6] "China’s CCTV surveillance network took just 7 minutes to capture BBC reporter – TechCrunch", TechCrunch, 2019. [Online]. Available: https://techcrunch.com/2017/12/13/china-cctv-bbc-reporter/. [Accessed: 15- Jul- 2019].

[7] Cohen, Neil, Jay Gattuso, and Ken MacLennan-Brown. CCTV operational requirements manual 2009. Home Office Scientific Development Branch, 2009.

[8] Flitton, Greg, Toby P. Breckon, and NajlaMegherbi. "A comparison of 3D interest point descriptors with application to airport baggage object detection in complex CT imagery." Pattern Recognition 46, no. 9 (2013): 2420-2436.