**UNUSUAL HUMAN ACTIVITY DETECTION USING OPEN CV PYTHON WITH MACHINE LEARNING**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO.** |
| **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **ABSTRACT** | vi  vii  xii  xiii |
| **CHAPTER 1: INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   1.4 PROPOSED SYSTEM  1.5 LITERATURE SURVEY | **1-2**  1  1  1  2  2 |
| **CHAPTER 2: PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.3 TECHNIQUE OR ALGORITHM | **6-7**  6  6  6  6  7 |
| **CHAPTER 3: REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS  3.4 Functional requirements  3.5 NON-FUNCTIONAL REQUIREMENTS | **8-9**  8  8  8  9  9 |
| **CHAPTER 4: SYSTEM DESIGN**  **4.1 general**  **4.2 UML DIAGRAMS**  4.2.1 USE CASE diagram  4.2.2 CLASS DIAGRAM  4.2.3 OBJECT DIAGRAM  4.2.4 COMPONENT DIAGRAM  4.2.5 DEPLOYMENT DIAGRAM  4.2.6 SEQUENCE DIAGRAM  4.2.7 COLLABORATION DIAGRAM  4.2.8 STATE DIAGRAM  4.2.9 ACTIVITY DIAGRAM  4.2.10 DATA FLOW DIAGRAM  4.2.11 E-R DIAGRAM  **4.3 SYSTEM ARCHITECTURE** | **10-20**  10  10  10  11  12  12  13  14  15  16  17  18  19  20 |
| **CHAPTER 5: DEVELOPMENT TOOLS**  5.1 PYTHON  5.1.1 HISTORY OF PYTHON  5.1.2 IMPORTANCE OF PYTHON  5.1.3 FEATURES OF PYTHON  5.1.4 LIBRARIES USED IN PYTHON | **21-23**  21  21  21  22  23 |
| **CHAPTER 6: IMPLEMENTATION**  6.1 GENERAL  6.1.1 CODING | **24**  24  24 |

|  |  |
| --- | --- |
| **CHAPTER 7: SNAPSHOTS**  7.1 GENERAL  7.1.1 VARIOUS SNAPSHOTS | **28**  28  28 |
| **CHAPTER 8: SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING  8.3.1 UNIT TESTING  8.3.2 FUNCTIONAL TEST  8.3.3 SYSTEM TEST  8.3.4 PERFORMANCE TEST  8.3.5 INTEGRATION TESTING  8.3.6 ACCEPTANCE TESTING  8.3.7 BUILD THE TEST PLAN | **29-31**  29  29  29  29  30  30  30  30  31  31 |
| **CHAPTER 9:**  9.1 APPLICATIONS  9.2 FUTURE ENHANCEMENT | **33**  33  33 |
| **CONCLUSION**  **REFERENCES** | **34**  **35** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.1 | ResNet Architecture | 7 |
| 4.1 | Use Case Diagram | 10 |
| 4.2 | Class Diagram | 11 |
| 4.3 | Object diagram | 11 |
| 4.4 | Component diagram | 12 |
| 4.5 | Deployment diagram | 13 |
| 4.6 | Sequence diagram | 13 |
| 4.7 | Collaboration Diagram | 14 |
| 4.8 | State Diagram | 15 |
| 4.9 | Activity Diagram | 16 |
| 4.10 | Data Flow Diagram | 17 |
| 4.11 | E-R Diagram | 18 |
| 4.12 | System Architecture | 19 |
| 5.1 | Python Libraries | 22 |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | NAME  Class B  Class A    Class B  Class A | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single class. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the process. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interaction between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communicates. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | ResNet | Residual Neural Network |
| 2. | ReLu | Rectified Linear Unit |
| 3. | GPL | General Public License |
| 4. | GNU | GNU’s not Unix |
| 5. | PERL | Practical extraction and report language |
| 6. | PHP | Hypertext Preprocessor |
| 7. | GUI | Graphical User Interface |
| 8. | Numpy | Numerical Python |

**ABSTRACT**

Now a day’s human behavior and activity pattern researches are more important in surveillance. Detection and tracking the object of behavior is important factor in video surveillance system. If any problem is happening in crowded area based on behaviors of persons then it depends on two types spatial and temporal. Over a last decade it has been seen the rapid growth and an extraordinary improvement in real-time video analysis. Main goal of video analytics is to identify the potential threaten events with less or no human intervention. Video surveillance is a prominent area of research which includes recognition of human activities and categorization of them into usual (normal), unusual (abnormal) or suspicious activities. Main task is to locating unusual events in videos by using some surveillance system which can be manual, semi-automatic or fully automatic. Manual surveillance system is fully dependent on human. It required manual labor to analyze behavior or to make difference between abnormal and normal behavior. Semiautomatic system required less human intervention while fully automatic are intelligent and smart video surveillance system which doesn’t required human intervention to make decision. The other method of intrusion detection is face recognition. The dataset of criminals is created and stored in system, when criminal face is recognized by camera it will create alert message to system and notify about it. Face recognition is done by OpenCV library in python. Internally image processing and deep learning is done in this process of recognition. Because of such advance technique system become more accurate.

**CHAPTERS**

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Simple activities can be modeled accurately as Markov Chains. However, complex or unfamiliar activities are often difficult to understand and model. For example, a researcher studying activities of daily living for a person with dementia will have a difficult time fitting a model unless she is an expert in dementia and understands its related behavioral science.

* 1. **OBJECTIVE**

1. The goal of activity recognition is to recognize common human activities in real life settings.
2. Accurate activity recognition is challenging because human activity is complex and highly diverse.
3. Several probability-based algorithms have been used to build activity models.
   1. **EXISTING SYSTEM**
4. Numerous attempts have been made in this field to automatize video surveillance but each and every approach has its own pros and cons.
5. On the basis of prior knowledge and human involvement in the learning process, the research in human activity recognition can be categorized as supervised, unsupervised and semi supervised.
   1. **PROPOSED SYSTEM**
6. Unusual activity recognition systems are developed to make surveillance system smarter and more intelligent.
7. Main aim is to detect suspicious or abnormal activities in videos to avoid future happening or to give alert whenever any type of mishappening occur.
8. These anomalous activity recognition systems classify normal and abnormal activities of objects. Most of previous research in anomalous or suspicious activity recognition has focused on behavior understanding by training the system manually.
9. Some of work shows unsupervised learning methodologies for activity detection.
   1. **LITERATURE SURVEY:**
      1. **DETECTING ACTIVITIES OF DAILY LIVING WITH SMART METER**

**Author:** J. Clement, J. Ploennigs, and K. Kabitzsch.

**Year:** 2013

**Description:** Smart meters provide us new information to visualize, analyze, and optimize the energy consumption of buildings, to enable demand-response optimizations, and to identify the usage of appliances. They also can be used to help older people to stay more independent in their homes by detecting their activity and their behavior models to ensure their healthy level. This paper reflects methods that can be used to analyze smart meter data to monitor human behavior in single apartments. Two approaches are explained in detail. The Semi-Markov-Model (SMM) is used to train and detect individual habits by analyzing the SMM to find unique structures representing habits. The distribution of the most possible executed activity (PADL) will be calculated to allow an evaluation of the currently executed activity (ADL) of the inhabitant. The second approach introduces an impulse based method that also allows the detection of ADLs and focuses on a temporal analysis of parallel ADLs. Both methods are based on smart meter events describing which home appliance was switched. Thus, this paper will also give an overview of popular strategies to detect switching events on electricity consumption data.

* + 1. **INCREMENTALLY MINING USAGE CORRELATIONS AMONG APPLIANCES IN SMART HOMES**

**Author:** Y. C. Chen, H. C. Hung, B. Y. Chiang

**Year:** 2015

**Description:** Recently, due to the great advent of sensor technology, residents can collect household appliance usage data easily. However, in general, usage data are generated progressively, visualizing how appliances are used from huge amount of data is challenging. Thus, an algorithm is needed to incrementally discover appliance usage patterns. Prior studies on usage pattern discovery are mainly focused on mining patterns while ignoring the incremental maintenance of mined results. In this paper, a novel method, Dynamic Correlation Miner (DC Miner), is developed to incrementally capture and maintain the usage correlations among appliances in a smart home environment. Furthermore, several optimization techniques are proposed to effectively reduce the search space. Experimental results indicate that the proposed method is efficient in execution time and possesses great scalability. Subsequent application of DC Miner on a real dataset also demonstrates its practicability.

* + 1. **THE ELDERLY ‘S INDEPENDENT LIVING IN SMART HOMES: A CHARACTERIZATION OF ACTIVITIES AND SENSING INFRASTRUCTURE SURVEY TO FASCILITATE SERVICES DEVELOPEMENT**

**Author:** Q. Ni, A. B. Garca Hernando

**Year:** 2015

**Description:** Human activity detection within smart homes is one of the basis of unobtrusive wellness monitoring of a rapidly aging population in developed countries. Most works in this area use the concept of "activity" as the building block with which to construct applications such as healthcare monitoring or ambient assisted living. The process of identifying a specific activity encompasses the selection of the appropriate set of sensors, the correct preprocessing of their provided raw data and the learning/reasoning using this information. If the selection of the sensors and the data processing methods are wrongly performed, the whole activity detection process may fail, leading to the consequent failure of the whole application. Related to this, the main contributions of this review are the following: first, we propose a classification of the main activities considered in smart home scenarios which are targeted to older people's independent living, as well as their characterization and formalized context representation; second, we perform a classification of sensors and data processing methods that are suitable for the detection of the aforementioned activities. Our aim is to help researchers and developers in these lower-level technical aspects that are nevertheless fundamental for the success of the complete application.

* + 1. **SMART-ENERGY GROUP ANAMOLY BASED BEHAVIORAL ABNORMALITY DETECTION**

**Author:** Alam, Roy, Petruska, and Zemp.

**Year:** 2016

**Description:** Monitoring behavioral abnormality of individuals living independently in their own homes is a key issue for building sustainable healthcare models in smart environments. While most of the efforts have been directed towards building ambient and wearable sensors-assisted activity recognition based behavioral analysis models for remote health monitoring, energy analytics assisted behavioral abnormality prediction have rarely been investigated. In this paper, we propose a data analytic approach that helps detect energy usage anomalies corresponding to the behavioral abnormality of the residents. Our approach relies on detecting everyday appliances usage from smart meter and smart plug data traces in regular activity days and then learning the unique time segment group of each appliance's energy consumption. We focus on detecting behavioral anomalies over a set of energy source data points rather than pinpointing individual odd points. We employ hierarchical probabilistic model-based group anomaly detection to interpret the anomalous behavior and therefore, detect potential tendency towards behavioral abnormality. We apply daily activity logs to evaluate our approach using two real world energy datasets pertaining to staged functional behaviors, and show that it is possible to detect max. 97% of anomalous days with max. 87% of meaningful micro-behavioral abnormal events generating 1.1% of false alarms. However, we show that our detected abnormality can be meaningfully represented to different stakeholders such as caregivers and family members to understand the nature and severity of abnormal human behavior for sustaining better healthcare.

* + 1. **CLOUD-SUPPORTED CYBER-PHYSICAL LOCALIZATIOB FRAMEWORK FOR PATIENTS MONITORING**

**Author:** M. S. Hossain

**Year:** 2017

**Description:** The potential of cloud-supported cyber-physical systems (CCPSs) has drawn a great deal of interest from academia and industry. CCPSs facilitate the seamless integration of devices in the physical world (e.g., sensors, cameras, microphones, speakers, and GPS devices) with cyberspace. This enables a range of emerging applications or systems such as patient or health monitoring, which require patient locations to be tracked. These systems integrate a large number of physical devices such as sensors with localization technologies (e.g., GPS and wireless local area networks) to generate, sense, analyze, and share huge quantities of medical and user-location data for complex processing. However, there are a number of challenges regarding these systems in terms of the positioning of patients, ubiquitous access, large-scale computation, and communication. Hence, there is a need for an infrastructure or system that can provide scalability and ubiquity in terms of huge real-time data processing and communications in the cyber or cloud space. To this end, this paper proposes a cloud-supported cyber-physical localization system for patient monitoring using smartphones to acquire voice and electroencephalogram signals in a scalable, real-time, and efficient manner. The proposed approach uses Gaussian mixture modeling for localization and is shown to outperform other similar methods in terms of error estimation.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Human activity understanding encompasses activity recognition and activity pattern discovery. The first focuses on accurate detection of the human activities based on a predefined activity model. Therefore, an activity recognition researcher builds a high-level conceptual model first, and then implements the model by building a suitable pervasive system.

**2.2 METHODOLOGIES**

**2.2.1 MODULES NAME**

* **Video Streaming**
* **Import model**
* **Normalization**
* **Activity Detection**

**2.2.2 MODULES EXPLANATION**

**Video streaming**

In this process we are capturing video from video devices and that video stream we are uploading into a Resnet model. Then the processing was started as detecting the activity.

**Import model**

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer visions tasks. he fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully.

**Normalization**

Image Normalization is a process in which we change the range of pixel intensity values to make the image more familiar or normal to the senses, hence the term normalization. Often image normalization is used to increase contrast which aids in improved feature extraction or image segmentation.

**ACTIVITY DETECTION**

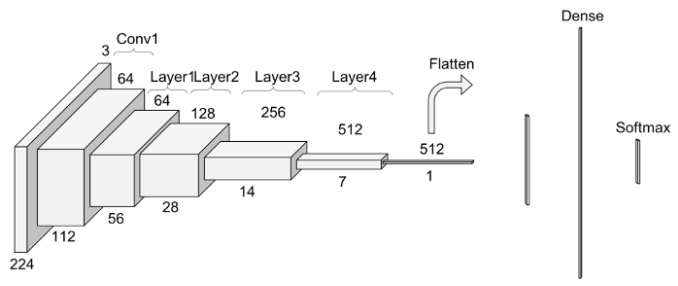
Then finally using Resnet detecting the activity of the person which passing on the video. Here based on the human behavior system detecting the activities of that person.

**2.3 ALGORITHM**

**ResNet**

When ResNet was first introduced, it was revolutionary for proving a new solution to a huge problem for deep neural networks at the time: the [vanishing gradient problem](https://www.youtube.com/watch?v=SKMpmAOUa2Q&t=6s). Although neural networks are universal function approximators, at a certain threshold adding more layers makes training become slower and makes the accuracy saturate. For instance, ResNet on the paper is mainly explained for ImageNet dataset.

So, let’s explain this repeating name, block. Every layer of a ResNet is composed of several blocks. This is because when ResNets go deeper, they normally do it by increasing the number of operations within a block, but the number of total layers remains the same - 4. An operation here refers to a convolution a batch normalization and a ReLU activation to an input, except the last operation of a block, that does not have the ReLU.



**Fig 2.1 ResNet Architecture**

**CHAPTER 3**

**REQUIREMENTS ANALYSIS**

**3.1 GENERAL**

On the other hand, activity pattern discovery is more about finding some unknown patterns directly from low-level sensor data without any predefined models or assumptions. Hence, the researcher of activity pattern discovery builds a pervasive system first and then analyzes the sensor data to discover activity patterns.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* Operating System : Windows 7/8/10
* Platform : Spyder3
* Programming Language : Python
* Front End : Spyder3

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their multi-modal topics, but also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed ETM model can exploit the multi-modal property of social event, which can effectively model social media documents including long text with related images and learn the correlations between textual and visual modalities to separate the visual-representative topics and non-visual-representative topics.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.2 UML Diagrams**

**4.2.1 USE CASE DIAGRAM**

User

Input video

Implementing model

Normalization

Feature extraction

Activity detection

Prediction

**Fig.4.1 Use Case Diagram**

**EXPLANATION**

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. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**4.2.2 CLASS DIAGRAM**

Normalization



Normalize data



Removenoisy ()



Removenull()

Feature extraction



Extract features



Applyalgorithm()

Prediction



Activity detection

User



Input video



Getdata()

**Fig 4.2.Class Diagram**

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project

**4.2.3 OBJECT DIAGRAM**

Normalization

Feature extraction

Prediction

User

**Fig.4.3 Object Diagram**

**EXPLANATION**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**4.2.4 COMPONENT DIAGRAM**

Input video

Implementing

Model

Normalization

Feature extraction

Activity detection

Prediction

**Fig.4.4 Component Diagram**

**EXPLANATION**

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

An assembly connector is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port."

When using a component diagram to show the internal structure of a component, the provided and required interfaces of the encompassing component can delegate to the corresponding interfaces of the contained components.

**4.2.5 DEPLOYMENT DIAGRAM**



**Fig.4.5 Deployment Diagram**

**4.2.6 SEQUENCE DIAGRAM**

**Fig.4.6 Sequence Diagram**

**EXPLANATION**

A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a message sequence chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.

**4.2.7 COLLABORATION DIAGRAM**

****

**Fig.4.7 Collaboration Diagram**

**EXPLANATION**

A collaboration diagram shows the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram. It can be the decomposition of a use case, use case diagram, or part of a use case diagram. The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

**4.2.8 STATE DIAGRAM**

input video

Implementing model

normalization

Feature extraction

Activity detection

prediction

**Fig.4.8 State Diagram**

**EXPLANATION**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.2.9 ACTIVITY DIAGRAM**



**Fig.4.9 Activity Diagram**

**EXPLANATION**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**4.2.10 DATA FLOW DIAGRAM**

Level 0

Dataset

Preprocessing

Human Activity Prediction

Splitting

Apply Algorithm

Level 1

Training Data, Testing Data

Input Data Set

Preprocessing

Human Activity Prediction

Data, Cleaning Data, Remove Null Values

Apply Algorithm

Splitting

Detection

**Fig.4.10 Data Flow Diagram**

**EXPLANATION**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**4.2.11 E-R DIAGRAM**

Preprocessing

Splitting

Human Activity

User

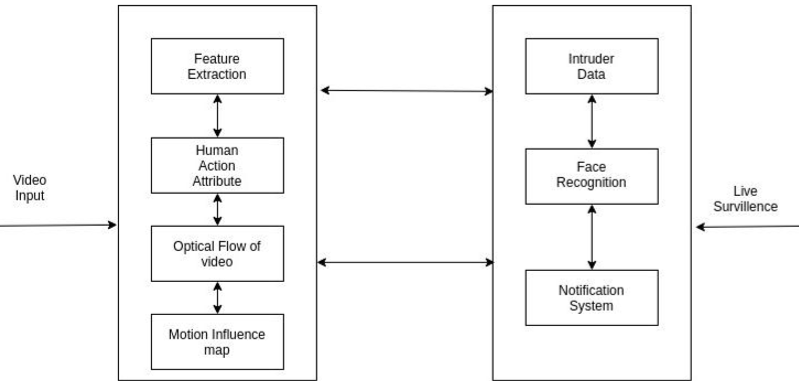
Algorithm

**Fig.4.11 E-R Diagram**

**EXPLANATION**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**4.3 SYSTEM ARCHITECHTURE**



**Fig.4.12 System Architecture**

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 PYTHON**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## 5.1.1 HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### 5.1.2 IMPORTANCE OF PYTHON

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### 5.1.3 FEATURES OF PYTHON

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.1.4 LIBRARIES USED IN PYTHON**

* NumPy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as data frames.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



**Fig.5.1 Python Libraries**

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**6.1.1 CODING**

# import the necessary packages

from collections import deque

import numpy as np

import imutils

import cv2

# load the contents of the class labels file, then define the sample

# duration (i.e., # of frames for classification) and sample size

# (i.e., the spatial dimensions of the frame)

CLASSES = open('action\_recognition\_kinetics.txt').read().strip().split("\n")

need = open('need.txt').read().strip().split("\n")

SAMPLE\_DURATION = 16

SAMPLE\_SIZE = 112

# initialize the frames queue used to store a rolling sample duration

# of frames -- this queue will automatically pop out old frames and

# accept new ones

frames = deque(maxlen=SAMPLE\_DURATION)

# load the human activity recognition model

print("[INFO] loading human activity recognition model...")

net = cv2.dnn.readNet('resnet-34\_kinetics.onnx')

# grab a pointer to the input video stream

print("[INFO] accessing video stream...")

vs = cv2.VideoCapture('vid.mp4')

# loop over frames from the video stream

while True:

# read a frame from the video stream

(grabbed, frame) = vs.read()

# if the frame was not grabbed then we've reached the end of

# the video stream so break from the loop

if not grabbed:

print("[INFO] no frame read from stream - exiting")

break

# resize the frame (to ensure faster processing) and add the

# frame to our queue

frame = imutils.resize(frame, width=400)

frames.append(frame)

# if our queue is not filled to the sample size, continue back to

# the top of the loop and continue polling/processing frames

#if len(frames) < SAMPLE\_DURATION:

#continue

# now that our frames array is filled we can construct our blob

blob = cv2.dnn.blobFromImages(frames, 1.0,

(SAMPLE\_SIZE, SAMPLE\_SIZE), (114.7748, 107.7354, 99.4750),

swapRB=True, crop=True)

blob = np.transpose(blob, (1, 0, 2, 3))

blob = np.expand\_dims(blob, axis=0)

# pass the blob through the network to obtain our human activity

# recognition predictions

net.setInput(blob)

outputs = net.forward()

label = CLASSES[np.argmax(outputs)]

print(label)

if label in need:

#raw the predicted activity on the frame

cv2.rectangle(frame, (0, 0), (300, 40), (0, 0, 0), -1)

cv2.putText(frame, label, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,

0.8, (255, 255, 255), 2)

# display the frame to our screen

cv2.imshow("Activity Recognition", frame)

key = cv2.waitKey(1) & 0xFF

# if the `q` key was pressed, break from the loop

if key == ord("q"):

break

cv2.destroyAllWindows()

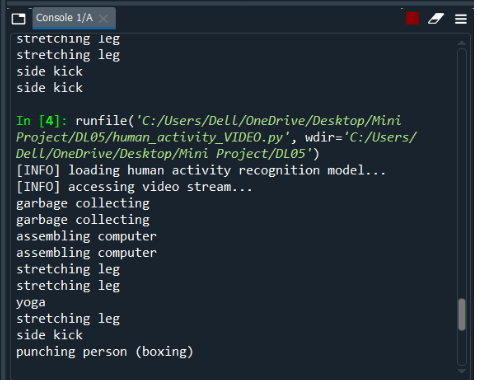
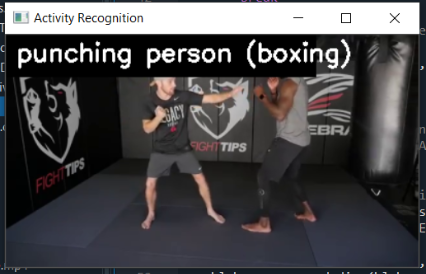
**CHAPTER 7**

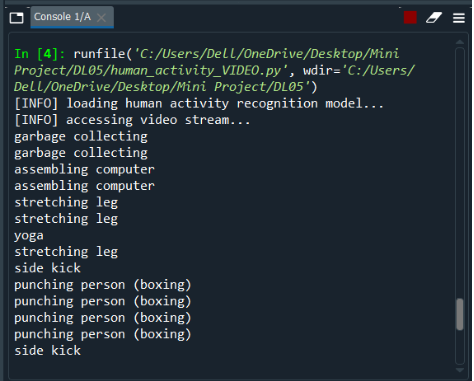
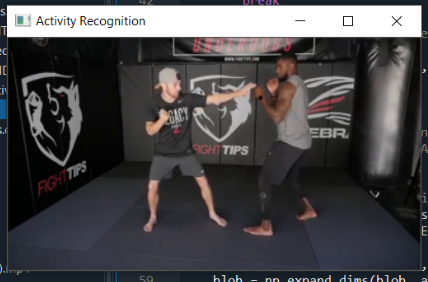
**SNAPSHOTS**

**7.1 GENERAL**

**7.1.1 VARIOUS SNAPSHOT**

**ACTIVITY RECOGNITION CONSOLE OUTPUT**

****

****

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

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 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3TYPES OF TESTS**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : Identified classes of valid input must be accepted.

Invalid Input : Identified classes of invalid input must be rejected.

Functions : Identified functions must be exercised.

Output : Identified classes of application outputs must be exercised.

Systems/Procedures : Interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.3.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No** | **Test scenario** | **Action** | **Expected result** | **Actual Result** | **Remarks** |
| 1. | Login | Verifying the credentials | Enter UserId and Password | Successfully Login. | Pass |
| 2. | Upload the leaf | Provide input image | Loading the input image from the system. | Successfully uploaded the image. | Successful |
| 3. | Preprocessing | Converting image into binary format. | Input image is converted into pixels. | Successfully converted. | Successful |
| 4. | Morphological Operations | After Conversion applying erosion and dilation | Remove unwanted pixels in the image. | Successfully removed unwanted pixels. | Successful |
| 5. | Segmentation | Convert the pixels into arrays | Image is segmented into parts. | Successfully segmented. | Successful |
| 6. | Features Extraction | Verifying the disease features | Disease features are verified for input image. | Successfully verified the features. | Successful |
| 7. | Disease Detection | Classifying the disease | By the features disease is detected. | Successfully disease detected for the uploaded leaf. | Successful |

**CHAPTER 9**

**9.1 APPLICATIONS**

* The face recognition using deep learning and image processing is used to detect the criminal in particular area such as bank, atm, public places etc.
* Training and monitoring a new employee to correctly perform a task (ex., proper steps and procedures when making a pizza, including rolling out the dough, heating oven, putting on sauce, cheese, toppings, etc.).
* Verifying that a food service worker has washed their hands after visiting the restroom or handling food that could cause cross-contamination (i.e., chicken and salmonella).
* Monitoring bar/restaurant patrons and ensuring they are not over-served.

**9.2 FUTURE ENHANCEMENT**

The Future work of the project is to investigate how injecting external knowledge would improve the results. While lexicons are a good way of doing that, as shown by Kiritchenko *et al.,* we are especially interested in exploiting more semantic alternatives, like ontologies or other semantic networks. Also, as we are dealing with unbalanced data, we plan to explore machine learning techniques that address this problem.

**CONCLUSION**

Human activity understanding encompasses activity recognition and activity pattern discovery. The system focuses on accurate detection of the human activities based on a predefined activity model. Therefore, an activity recognition researcher builds a high-level conceptual model, and then implements the model by building a suitable pervasive system. On the other hand, activity pattern discovery is more about finding some unknown patterns directly from data without any predefined models or assumptions.

**REFERENCES**

[1] Alam, Roy, Petruska, and Zemp, “Smart-energy group anomaly based behavioral abnormality detection,” in 2016 IEEE Wireless Health (WH), Oct 2016.

[2] C. Chalmers, W. Hurst, M. Mackay, and P. Fergus, “Smart meter profiling for health applications,” in 2015 International Joint Conference on Neural Networks (IJCNN), July 2015.

[3] J. Clement, J. Ploennigs, and K. Kabitzsch, Detecting Activities of Daily Living with Smart Meters, ch. Advance Technology and Societal Change Springer, Germany, 11 2013.

[4] J. Liao, L. Stankovic, and V. Stankovic, “Detecting household activity patterns from smart meter data,” in Intelligent Environments (IE), 2014 International Conference on, 6 2014.

[5] K. Gajowniczek and T. Zabkowski, “Data mining techniques for detecting household characteristics based on smart meter data,” Energies, vol. 8, no. 7,2015.

[6] Marc Wenninger, Jochen Schmidt and Toni Goeller, “Appliance Usage Prediction for the Smart Home with an Application to Energy Demand Side management and Why Accuracy is not a Good Performance Metric for this Problem”, SMARTGREENS 2017 - 6th International Conference on Smart Cities and Green ICT Systems.

[7] M. S. Hossain, “Cloud-supported cyber-physical localization framework for patients monitoring,” IEEE Systems Journal, vol. 11, no. 1, pp. 118–127, March 2017.

[8] Q. Ni, A. B. Garca Hernando, and I. P. de la Cruz, “The elderly’s independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development”,2015.

[9] Y. C. Chen, H. C. Hung, B. Y. Chiang, S. Y. Peng, and P. J. Chen, “Incrementally mining usage correlations among appliances in smart homes,” in Network-Based Information Systems (NBiS), 2015 18th International Conference on, 2015.