AI DRIVEN DIAGNOSTIC CENTER SUPPORT FOR DETECTION OF CHRONIC DISEASES

A Project report submitted in partial fulfillment of requirements for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE IN
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING(CSM)

By

ADIKE ARUN KUMAR (219X1A3333)

Under the esteemed guidance of
Smt. K. ASHA RANI
Assistant Professor
Department of E.C.S

Department of Emerging Technologies in Computer Science

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL
(Affiliated to JNTUA, ANANTAPUR)

2024 – 2025

Department of Emerging Technologies in Computer Science

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTAPUR)

CERTIFICATE

This is to certify that the Project Work entitled 'AI DRIVEN DIAGNOSTIC CENTER SUPPORT FOR DETECTION OF CHRONIC DISEASES' **is a bonafide record of work carried out by**

ADIKE ARUN KUMAR (219X1A3333)

Under my guidance and supervision in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Smt.K.ASHA RANI
AssistantProfessor,
Department of ECS.,
G. Pulla Reddy Engineering College,
Kurnool.

Dr. R. Praveen Sam
Professor & Head of the Department,
Department of ECS.,
G.Pulla Reddy Engineering College,
Kurnool.

Signature of the External Examiner : .....

DECLARATION

I hereby declare that the project titled "AI DRIVEN DIAGNOSTIC CENTER SUPPORT FOR DETECTION OF CHRONIC DISEASES" is an authentic work carried out by me as the student of **G. PULLA REDDY ENGINEERING COLLEGE (Autonomous) Kurnool,** during 2024-25 and has not been submitted else where for the award of any degree or diploma in part or in full to any institute.

ADIKE ARUN KUMAR
(219X1A3333)

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude to my project guide **Smt. K. ASHA RANI**, **Assistant Professor** of Emerging Technologies in Computer Science Department, G. Pulla Reddy Engineering College, for his/her immaculate guidance, constant encouragement and cooperation which have made possible to bring out this project work.

I am grateful to my project in charge **Smt. K. ASHA RANI, Assistant Professor** of Emerging Technologies in Computer Science Department, G. Pulla Reddy Engineering College, for helping me and giving me the required information needed for our project work.

I am thankful to my Head of the Department **Dr. R. Praveen Sam Garu**, for his whole hearted support and encouragement during the project sessions.

I am grateful to my respected Principal **Dr. B. Sreenivasa Reddy Garu,** for providing requisite facilities and helping us in providing such a good environment.

I wish to convey my acknowledgements to all the staff members of the Emerging Technologies in Computer Science department for giving the required information needed for our project work.

Finally, I wish to thank all my friends and well wishers who have helped me directly or indirectly during the course of this project work.

ABSTRACT

Artificial intelligence (AI) has the potential to significantly improve patient outcomes and diagnostic accuracy in the quickly developing sector of healthcare. The goal of this project is to create an AI- powered diagnostic assistance system that will help medical practitioners identify chronic illnesses including cancer, heart disease, and diabetes at an early stage. In order to find patterns suggestive of an early disease onset, the system will use sophisticated machine learning algorithms to evaluate patient data, including medical history, test results, and imaging data. The system will offer actionable insights and risk assessments to improve decision-making and enable prompt intervention by fusing real-time data processing and predictive analytics. To achieve high accuracy and generalizability, the research will concentrate on building a solid model that has been trained on a variety of datasets. To improve accessibility and expedite productivity, the system will also be built to seamlessly interact with current electronic health record (EHR) systems. The anticipated outcome is a tool that helps to enhance patient care and outcomes by lowering the workload of medical practitioners and increasing diagnostic precision.

.

CONTENTS

LIST OF FIGURES

INTRODUCTION

1. INTRODUCTION

Diagnosing chronic diseases, such as diabetes, cardiovascular issues, and respiratory disorders, presents a considerable challenge due to the intricate nature of symptoms and disease progression. Traditional diagnostic approaches, though effective, often involve labor-intensive manual analysis and interpretation, which can be susceptible to errors and inefficiencies. With the increasing incidence of chronic conditions worldwide, there is an urgent need for innovative methods to enhance the accuracy and efficiency of diagnostics.

1.1 INTRODUCTION

Artificial intelligence (AI) has emerged as a pivotal technology in healthcare, offering advanced solutions to support and improve diagnostic processes. AI-driven diagnostic support systems employ sophisticated Machine Learning (ML) and Deep Learning (DL) techniques to analyze large volumes of medical data with greater precision. These systems are capable of automating the evaluation of medical images, detecting intricate patterns, and generating predictions, which helps reduce the burden on healthcare professionals and enhances diagnostic performance.

Machine Learning encompasses a variety of techniques, including neural networks and fuzzy logic, which contribute to automating and optimizing diagnostic workflows. These methods enable AI systems to learn from data and continually improve their predictive accuracy. Deep Learning, a specialized area within ML, utilizes Convolutional Neural Networks (CNNs) to conduct advanced image analysis without the need for manual feature extraction, leading to more accurate detection and classification of disease-related features in medical images.

The application of AI in chronic disease detection has the potential to significantly transform healthcare by providing more accurate and timely diagnosis, facilitating early intervention, and improving overall patient care. However, challenges remain, such as issues with data quality, algorithm transparency, and integration into existing healthcare practices. Addressing these challenges is crucial for maximizing the benefits of AI-driven diagnostic systems.

A detailed review of AI-driven diagnostic support systems for chronic disease detection, highlighting recent technological advancements, evaluating their effectiveness, and exploring current limitations. By examining the impact of AI on chronic disease diagnostics and suggesting future research avenues, this review aims to advance the development and integration of AI technologies in healthcare.

1.2 MOTIVATION

Improved Patient Outcomes: Leveraging AI in diagnostics aims to enhance patient care by enabling early detection of chronic illnesses, improving prognosis through timely intervention.

Enhanced Diagnostic Accuracy: Advanced machine learning algorithms analyze complex data, reducing the likelihood of human error and increasing diagnostic precision.

Early Detection of Chronic Diseases: The system focuses on identifying diseases such as cancer, heart disease, and diabetes in their early stages, when treatment is most effective.

Comprehensive Data Analysis: By integrating patient medical history, test results, and imaging data, the system provides a holistic view for better diagnosis.

Actionable Insights for Decision-Making: The tool offers predictive analytics and risk assessments, enabling healthcare providers to make informed decisions swiftly.

Integration with EHR Systems: Seamless compatibility with electronic health record systems ensures widespread adoption and streamlined workflows in healthcare facilities.

Workload Reduction for Medical Professionals: Automating data analysis and diagnostics allows practitioners to focus on patient care, reducing their workload and preventing burnout.

Accessibility and Scalability: The system is designed to be accessible across various healthcare setups, ensuring its impact is broad and scalable.

Research-Driven Development: The project emphasizes training the model on diverse datasets to achieve high accuracy and generalizability, addressing biases and ensuring reliability.

Focus on Real-Time Processing: Incorporating real-time data processing ensures the system delivers timely insights critical for immediate clinical decisions.

Engaging User Experience: Aims to provide users with an engaging and entertaining experience that goes beyond conventional drawing applications.

1.3 PROBLEM DEFINITION

Chronic diseases such as cancer, heart disease, and diabetes are leading causes of death and disability globally, placing immense pressure on healthcare systems and economies. Early detection of these conditions is critical for successful treatment and improved patient outcomes. However, traditional diagnostic methods often fall short due to their reliance on manual processes that are time-consuming, error-prone, and limited in their ability to analyze complex and voluminous healthcare data. As a result, many chronic conditions are diagnosed at advanced stages, when treatment options are less effective, leading to higher mortality rates and increased healthcare costs. These challenges are compounded by the rising demand on healthcare professionals, who are often overwhelmed with large workloads, further increasing the likelihood of diagnostic delays and errors.

To address these pressing challenges, there is a need for an innovative diagnostic support system powered by artificial intelligence (AI). Such a system can leverage advanced machine learning algorithms to analyze diverse datasets, including medical histories, imaging studies, and laboratory results, with speed and accuracy far beyond human capabilities. The use of predictive analytics and real-time data processing will enable the system to detect early signs of chronic illnesses, providing actionable insights and risk assessments for medical practitioners. This capability not only enhances diagnostic accuracy but also facilitates timely interventions, significantly improving patient outcomes. Moreover, by automating the analysis of complex medical data, the system can alleviate the workload of healthcare professionals, allowing them to focus on patient care and decision-making.

A critical aspect of the proposed system is its seamless integration with existing electronic health record (EHR) systems. This compatibility ensures that the solution is accessible and scalable across diverse healthcare settings, enabling widespread adoption and impact. By streamlining workflows and providing reliable diagnostic support, the system can transform chronic disease management, making it more efficient and effective. Ultimately, this AI-driven solution has the potential to revolutionize healthcare by improving diagnostic precision, reducing the burden on healthcare practitioners, and delivering better outcomes for patients worldwide.

1.4 OBJECTIVE OF THE PROJECT

Problem statement

Chronic diseases such as cancer, heart disease, and diabetes are significant global health challenges due to their high prevalence, severe outcomes, and associated costs. Early detection is critical for improving patient outcomes, yet traditional diagnostic methods, reliant on manual analysis of complex patient data, are often time-consuming, error-prone, and inadequate for handling the growing volume of healthcare information. These limitations lead to delayed diagnoses, reduced treatment efficacy, and increased strain on healthcare systems. An AI-driven diagnostic support system is essential to address these issues by leveraging machine learning algorithms to analyze diverse datasets, identify early disease indicators, and provide actionable insights for timely interventions. By integrating seamlessly with existing electronic health record (EHR) systems, such a solution can enhance diagnostic accuracy, reduce medical practitioners' workloads, and transform chronic disease management for better patient care and outcomes.

Proposed System

The proposed system can be classified into mainly two steps after acquiring the input data from the user who are using our application and the other one is getting the result by processing the input data.. These steps are: Extraction Method and Features estimation and Extraction.

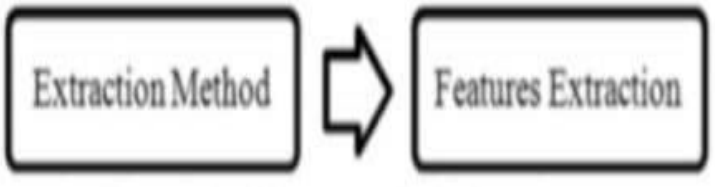

Fig 1.4.1 Steps of Extraction

1.5 ORGANIZATION OF THE REPORT

The report gives the reader a summary of the project and details the methodical execution of the developed working application. It also provides an overview of the project's potential for implementation.

Chapter 1: Introduction is about the AI DRIVEN DIAGNOSTIC CENTER SUPPORT FOR DETECTION OF CHRONIC DISEASES motivation, definition and objective of the project.

Chapter 2: System requirement specifies all the requirements that are needed for developing the application, which includes hardware and software requirements.

Chapter 3: Literature survey details about Chronic diseases detection using Advanced techniques and covers the reason behind developing the project.

Chapter 4: System Design and UML diagrams are shown.

Chapter 5: Entire source code and results of the Implementation are shown.

Chapter 6: All the testing strategies that are involved to test the model, has been described in this section.

Chapter 7: Future enhancement section provides the details about the extension of the project that are to be implemented in the future and what can be added in future.

Chapter 8: References

SYSTEM SPECIFICATIONS

2. SYSTEM SPECIFICATIONS

2.1 SOFTWARE SPECIFICATIONS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

The system should be able to interface with the existing system

- The system should be accurate
- The system should be better than the existing system

2.2 HARDWARE SPECIFICATIONS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list, especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture.

The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored.

HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS

- RAM : 8GB
- Processor : Intel ICore
- Hard disk : 2TB

SOFTWARE REQUIREMENTS

- Operating System : Windows
- Libraries : Numpy,Pandas, MatplotLib,Seaborn,Scikit Learn,Keras
- Framework : Flask
- Language : Python

LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 EXISTING SYSTEM

3.1.1 Evaluation of artifcial intelligence techniques in disease diagnosis and prediction

Authors: Nafseh Ghafar Nia1, Erkan Kaplanoglu1, Ahad Nasab1

Published Year: 2023

The research paper "Evaluation of Artificial Intelligence Techniques in Disease Diagnosis and Prediction" provides a comprehensive review of how artificial intelligence (AI) is revolutionizing medical diagnostics. It highlights the critical role of AI, particularly through machine learning (ML) and deep learning (DL), in automating disease detection and improving diagnostic accuracy. By processing complex medical images such as CT scans, X-rays, and MRIs, AI-based models like Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs) have significantly enhanced the efficiency of disease identification. These technologies reduce physician workloads, minimize errors, and improve early detection rates, especially for conditions such as cancers, cardiovascular diseases, and neurological disorders.

The paper emphasizes the wide-ranging applications of AI in the healthcare sector. For instance, DL models excel in analyzing large datasets and automatically extracting relevant features, making them particularly effective for medical imaging tasks such as segmentation, classification, and fusion. Predictive modeling is another area where AI has proven invaluable, as it helps forecast disease progression and identify at-risk individuals. The study also showcases specific implementations, such as AI frameworks for diagnosing Alzheimer’s disease, Parkinson’s disease, and breast cancer, achieving remarkable levels of accuracy that often surpass human expertise. AI technologies are also used for real-time monitoring of patients, offering personalized insights and enabling more effective health management.

However, the study also discusses significant challenges in implementing AI in medical diagnostics. A major hurdle is the reliance on large, labeled datasets for training AI models, which are not always readily available. Additionally, the complexity of DL architectures and the computational power they require pose significant barriers. Ethical concerns, such as maintaining patient privacy and addressing biases in AI algorithms, further complicate the integration of these technologies into mainstream healthcare. The paper suggests potential solutions, including the use

of data augmentation to expand training datasets and techniques like model compression to reduce computational demands.

Looking ahead, the paper envisions a future where AI becomes an integral part of healthcare, enhancing patient outcomes and optimizing clinical workflows. Advances in AI could lead to more precise diagnostics, personalized treatment plans, and early disease detection, significantly improving global healthcare systems. However, achieving this vision will require collaboration between AI researchers, medical professionals, and policymakers to address current limitations and ethical concerns. The study concludes by emphasizing that with continuous advancements, AI holds the promise of transforming disease diagnosis and prediction into a more efficient, accurate, and accessible process.

3.1.2 Artificial intelligence in disease diagnostics: A critical review and classification on the current state of research guiding future direction

Authors: Milad Mirbabaie , Stefan Stieglitz ,Nicholas R. J. Frick

Published Year: 2021

The research paper critically examines the role of artificial intelligence (AI) in medical diagnostics, providing an overview of its current applications and a classification of AI techniques used in healthcare. It highlights how AI can address diagnostic challenges arising from human error, cognitive overload, and time constraints in the medical field. AI systems, such as neural networks and decision trees, are shown to improve diagnostic accuracy, efficiency, and consistency by processing large datasets, integrating data from multiple sources, and applying advanced algorithms.

The paper explores various AI approaches like supervised, unsupervised, and deep learning, emphasizing their applicability to specific diseases and datasets. Supervised learning techniques, such as support vector machines and neural networks, have been widely used for disease prediction and classification. Deep learning, with its ability to process complex, multidimensional data, has been particularly impactful in fields like dermatology, cardiology, and oncology. Despite their promise, these methods often operate as "black boxes," raising concerns about explainability and trust in AI systems.

Addressing challenges like data availability, model transparency, and ethical considerations, the study proposes future research directions. It underscores the need for larger, more diverse datasets, better explainable AI models, and integration of AI into real-world healthcare settings. The paper concludes by suggesting collaborative efforts between AI developers, healthcare practitioners, and researchers to enhance AI's diagnostic capabilities and ensure its safe, effective adoption in clinical practice.

3.1.3 Applications of Machine Learning Predictive Models in the Chronic Disease Diagnosis

3.1.4 Sequences Authors: Gopi Battineni, Getu Gamo Sagaro, Nalini Chinatalapudi, Francesco Amenta

Published Year: 2020

The research paper explores the applications of machine learning (ML) predictive models in diagnosing chronic diseases (CDs). Chronic diseases contribute significantly to global healthcare costs, necessitating lifelong treatment. The study examines the capabilities of ML in early detection, diagnosis, and forecasting of chronic conditions, highlighting how predictive models improve patient outcomes and streamline medical decision-making. A systematic review of 453 articles from major medical databases filtered down to 22 studies demonstrates the strengths and limitations of various ML models. Algorithms such as Support Vector Machines (SVM), Logistic Regression (LR), and clustering methods are identified as widely used tools for accurate disease classification and prediction.

Key findings include the performance of different models in diagnosing CDs like diabetes, cardiovascular diseases, chronic obstructive pulmonary disease (COPD), and liver diseases. SVM and LR emerge as effective tools due to their high accuracy and reliability, while artificial neural networks (ANN) showcase their utility in recognizing patterns within complex datasets. The research highlights specific pathologies where these models excel, such as COPD exacerbation forecasting and diabetes classification. However, the study emphasizes the variability in outcomes due to the diversity of datasets and modeling approaches, pointing out that no single model can claim universal applicability across all medical conditions.

The paper also discusses challenges associated with adopting ML in clinical practice. Issues like the need for high-quality and diverse datasets, ethical concerns regarding patient data privacy, and the complexity of integrating predictive models into existing healthcare workflows are addressed. Furthermore, the study identifies limitations in the current use of supervised learning models and advocates for increased exploration of unsupervised and deep learning techniques to improve diagnostic precision. Emphasis is placed on developing standardized protocols to ensure the effectiveness and reliability of ML applications in medicine.

The authors conclude by underscoring the transformative potential of ML in advancing healthcare. They recommend fostering interdisciplinary collaboration among researchers, clinicians, and policymakers to enhance the adoption of AI-driven tools. With continuous advancements, predictive models can play a critical role in mitigating the burden of chronic diseases, optimizing resource allocation, and improving patient care outcomes. Future research directions include refining algorithms to handle complex medical imaging data and addressing gaps in AI implementation for various chronic conditions.

3.2 DISADVANTAGES OF EXISTING SYSTEMS

Although ML models show promise in terms of efficiency and accuracy in the diagnosis of chronic illnesses, a number of drawbacks were observed:

Data Dependency: AI models require large, diverse, and high-quality datasets for effective training, which are often unavailable or difficult to access due to privacy concerns and regulatory restrictions.

Model Explainability: Many AI algorithms, especially deep learning models, function as "black boxes," making it difficult for clinicians to trust or interpret their outputs.

Ethical and Regulatory Issues: Concerns about bias in training data, patient privacy, and compliance with ethical guidelines remain unresolved, potentially hindering widespread adoption

Lack of Standardization: The study indicates no standard approach for applying AI models across different diseases, leading to inconsistent results and reduced generalizability.

Computational Complexity: Advanced models like neural networks demand significant computational resources, making them less accessible to low-resource healthcare settings.

Trust Issues: Clinicians may find it challenging to adopt these technologies due to a lack of transparency in the decision-making process of complex models like neural networks.

Integration Challenges: Incorporating ML models into clinical workflows is complex and requires significant infrastructural changes, including retraining healthcare professionals.

Dataset Variability: Variations in dataset size, quality, and sources across studies result in inconsistent model performances and hinder reproducibility.

Data Quality: Predictions can be wrong if there is biased or insufficient data.

Interpretability of the Model: Healthcare professionals may find it difficult to trust

complex models due to their lack of transparency.

Generalization: Models developed for particular populations might not function

effectively for a variety of patient populations.

3.3 PROPOSED SYSTEM

1. Data Collection and Integration

- **Input Sources**: Medical history, laboratory test results, imaging data (e.g., X-rays, MRIs), and other patient-specific data will serve as inputs.
- **EHR Integration**: Seamless integration with existing electronic health record (EHR) systems to automatically fetch and store patient data for analysis.
- **Real-Time Updates**: Incorporation of real-time data processing to keep the system updated with the latest patient information.

2. Data Preprocessing

- **Data Cleaning**: Automated cleaning to handle missing values, inconsistent formats, and redundant entries.
- **Feature Extraction**: Identification of key diagnostic indicators from structured and unstructured data.
- **Anonymization**: Ensure patient privacy by anonymizing sensitive data in compliance with regulations such as HIPAA.

3. Machine Learning and Predictive Analytics

- **Model Development**:
    - Develop and train advanced machine learning models (e.g., deep neural networks, ensemble methods) on diverse and high-quality datasets.
    - Utilize a variety of data sources, including publicly available medical datasets and proprietary data from healthcare institutions.
- **Risk Scoring**: Generate personalized risk scores for chronic illnesses based on patient data patterns.
- **Early Detection**: Employ algorithms capable of identifying subtle patterns indicative of early disease onset.

4. Actionable Insights and Decision Support

- **Diagnostic Recommendations**: Provide healthcare professionals with diagnostic suggestions and probable conditions based on analyzed data.
- **Visualization**: Offer user-friendly data visualizations, including trend graphs and heatmaps, to aid interpretation.
- **Treatment Guidance**: Highlight potential treatment pathways and further diagnostic steps.

5. System Accessibility and Usability

- **User Interface**:
    - Develop an intuitive interface accessible to medical practitioners.
    - Include role-based access to accommodate different levels of users (e.g., doctors, technicians).
- **Mobile and Desktop Platforms**: Provide compatibility with multiple devices to ensure accessibility across healthcare settings.

6. Evaluation and Validation

- **Clinical Testing**: Validate the system's accuracy and reliability through rigorous clinical trials.
- **Performance Metrics**: Evaluate performance using metrics such as sensitivity, specificity, and

accuracy.

- **Feedback Mechanism**: Incorporate user feedback to iteratively improve system performance.

7. Security and Compliance

- **Data Security**: Implement encryption, secure authentication, and regular audits to safeguard sensitive patient data.
- **Regulatory Compliance**: Ensure compliance with healthcare regulations such as HIPAA, GDPR, and local guidelines.

8. Benefits and Impact

- **Enhanced Diagnostic Accuracy**: Reduce diagnostic errors and improve precision through advanced algorithms.
- **Reduced Workload**: Streamline workflows for medical practitioners, allowing them to focus on patient care.
- **Improved Patient Outcomes**: Enable earlier interventions, potentially reducing the severity and progression of chronic illnesses.

.

DESIGN

4. DESIGN

4.1 INTRODUCTION

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It's a crucial phase in the development of complex systems, whether they are software applications, hardware systems, or a combination of both. The primary goal of system design is to create a blueprint that guides the construction and implementation of the system. ensuring that it performs effectively, efficiently, and reliably while meeting the intended functionality and user requirements.

1. NumPy

NumPy (Numerical Python) is a core library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these structures. It is highly efficient and serves as the foundation for many other scientific libraries, making it essential for handling numerical data in data science and machine learning tasks. Commonly used for linear algebra, statistical operations, and data manipulation, NumPy simplifies complex numerical computations with its vectorized operations, offering significant speed improvements over standard Python.

2. Pandas

Pandas is a powerful Python library for data manipulation and analysis, introducing intuitive data structures like DataFrame and Series. It simplifies the process of handling structured data by providing robust tools for cleaning, filtering, aggregating, and transforming datasets. With extensive support for handling missing data and integrating with file formats like CSV, Excel, and SQL databases, Pandas is widely used for preparing and exploring data before analysis or modeling, making it a staple in data science workflows.

3. Matplotlib

Matplotlib is a versatile plotting library in Python for creating static, interactive, and animated visualizations. It offers fine-grained control over every aspect of a plot, from axis labels to legend placement, enabling users to create publication-quality figures. With support for various plot types such as line, bar, scatter, and histogram plots, Matplotlib is a go-to tool for visualizing trends,

distributions, and relationships in data, often used alongside other libraries like NumPy and Pandas.

4. Seaborn

Seaborn is a high-level visualization library built on top of Matplotlib, designed to make statistical graphics more attractive and informative. It provides built-in themes, color palettes, and functions to easily create complex visualizations like heatmaps, violin plots, and pair plots. Seamlessly integrating with Pandas DataFrames, Seaborn simplifies exploratory data analysis by highlighting patterns and relationships in data while enhancing the aesthetic appeal of plots with minimal effort.

5. Scikit-Learn

Scikit-Learn is a comprehensive machine learning library in Python that provides efficient implementations of a wide range of algorithms for classification, regression, clustering, and dimensionality reduction. It includes tools for data preprocessing, model selection, evaluation, and hyperparameter tuning, making it ideal for both beginners and experienced practitioners. With its simple and consistent API, Scikit-Learn is widely used for building, training, and evaluating machine learning models across various applications.

6. Keras

Keras is a high-level deep learning library that simplifies the process of building and training neural networks. Running on top of backends like TensorFlow, it provides an intuitive interface for defining complex architectures such as convolutional and recurrent neural networks. Keras is widely appreciated for its modular design and user-friendliness, making it an excellent choice for both research and production-level deep learning projects. It supports rapid prototyping, multi-GPU training, and deployment, enabling efficient experimentation with advanced AI models.

Algorithms used for Hand Tracking

Hand gesture tracking and recognition are handled by the MediaPipe framework, and computer vision is handled by the OpenCV package. The application tracks and recognizes

hand movements and hand tips using machine learning algorithms.

4.2 FLOW CHART OF THE MODEL

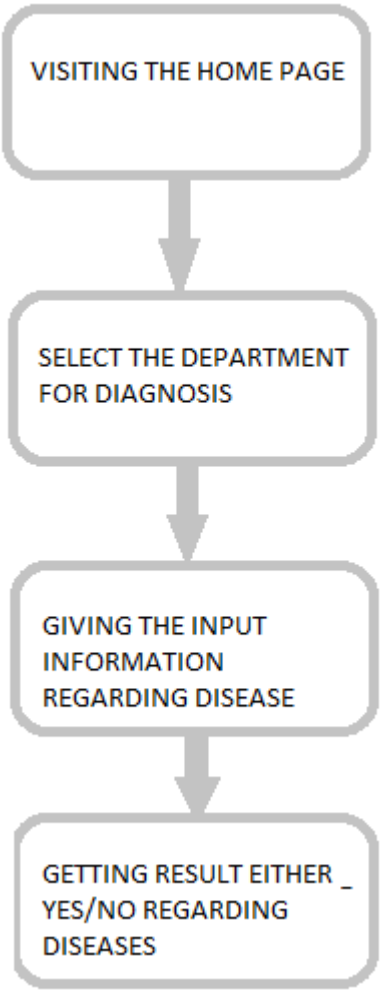

Fig- 4.2.1: Flow Chart

Creating a good User friendly interface

Developing the Web pages

The web pages that we developed are home page, services page, about us page, login page, register page. Here the key web page is services page where we included 3 departments for diagnosis they are Cardiology, Diabetes, Liver Diseases.

Detecting whether the person is affected by respective chronic disease

Import the Required Modules: Importing Numpy,Pandas, MatplotLib,Seaborn,Scikit Learn,Keras for data handling, data analysis, data visualization, selecting a best model for prediction and numerical calculation.

Installing Flask Framework: Flask is a lightweight Python web framework used for building web applications and RESTful APIs. It offers simplicity, flexibility, and extensibility, making it ideal for small projects, prototyping, or scalable applications. With features like Jinja2 templating and support for extensions, Flask provides full control for developers to customize their applications.

Data Collection and Data Preprocessing: We collected data from kaggle, data bank and some records manually. Data preprocessing is done to the data we collected in order to eliminate noise, over fitting and handling categorical data and ranging the values by MinMax and Standard Scaler.

Splitting Dataset into Training and Testing Data: Split the dataset into training and testing dataset in the ratio of 80 : 20 by train_test_split() in sklearn.model_selection

Initialize Machine Learning Model: Initialize a pre-trained machine learning model. Set the parameter for machine learning model like random_state, max_features, n_estimators, min_sample_leaf for Random Forest, learning_rate, n_epochs and random_state for Logistic Regression.

Evaluate Model Performance: Model performance has evaluated in which for regression we used r2_score and for classification we used Confusion Matrix(Random Forest and Logistic Regression).

Initiate app.py: Here we need to run the app.py file to initiate the respective department file functionality.

Give the inputs : Here we need to give inputs by selecting the respective departments.

since for every chronic diseases the factor or features may differ.

Output

Getting the result: The output is generated in the next page by showing where the person is affected by particular chronic disease or not.

Display the recorder coordinates

Displaying recorded coordinates typically involves visualizing the data on a graphical user interface (GUI) or within a plotting framework.

4.3 UML DIAGRAMS

4.3.1 USECASE DIAGRAM

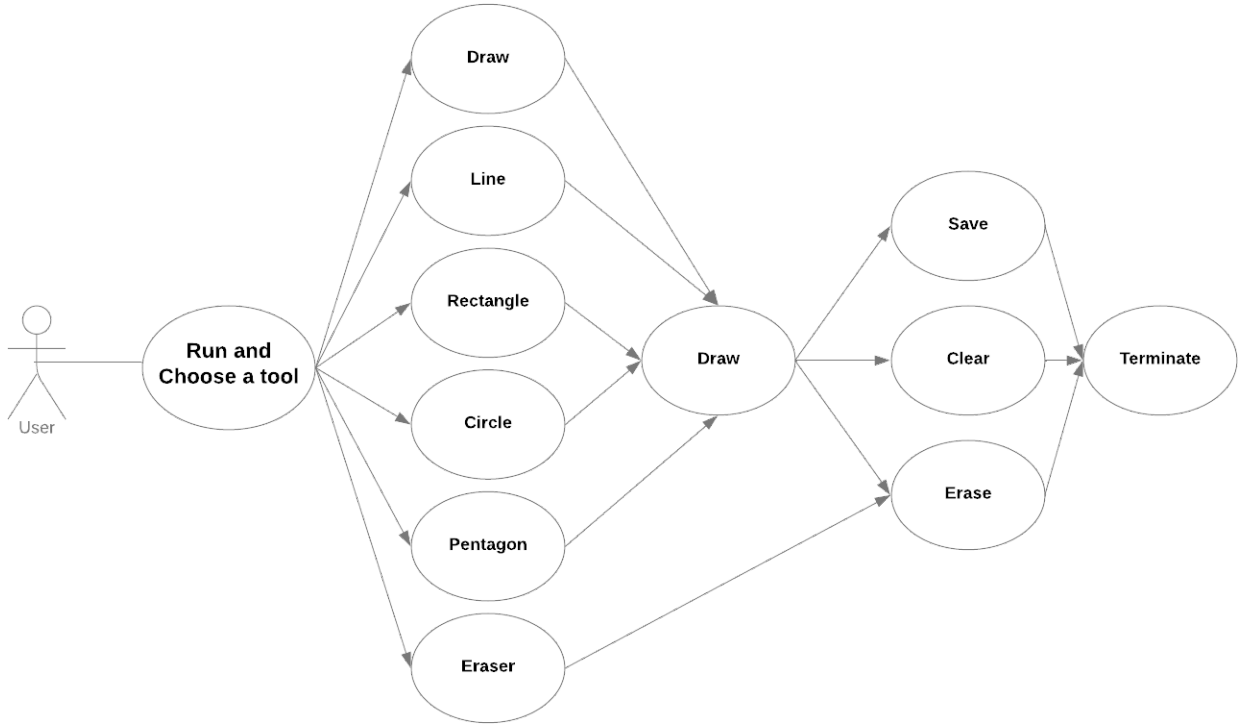

Fig-4.3.1: Use Case Diagram

4.3.2 ACTIVITY DIAGRAM

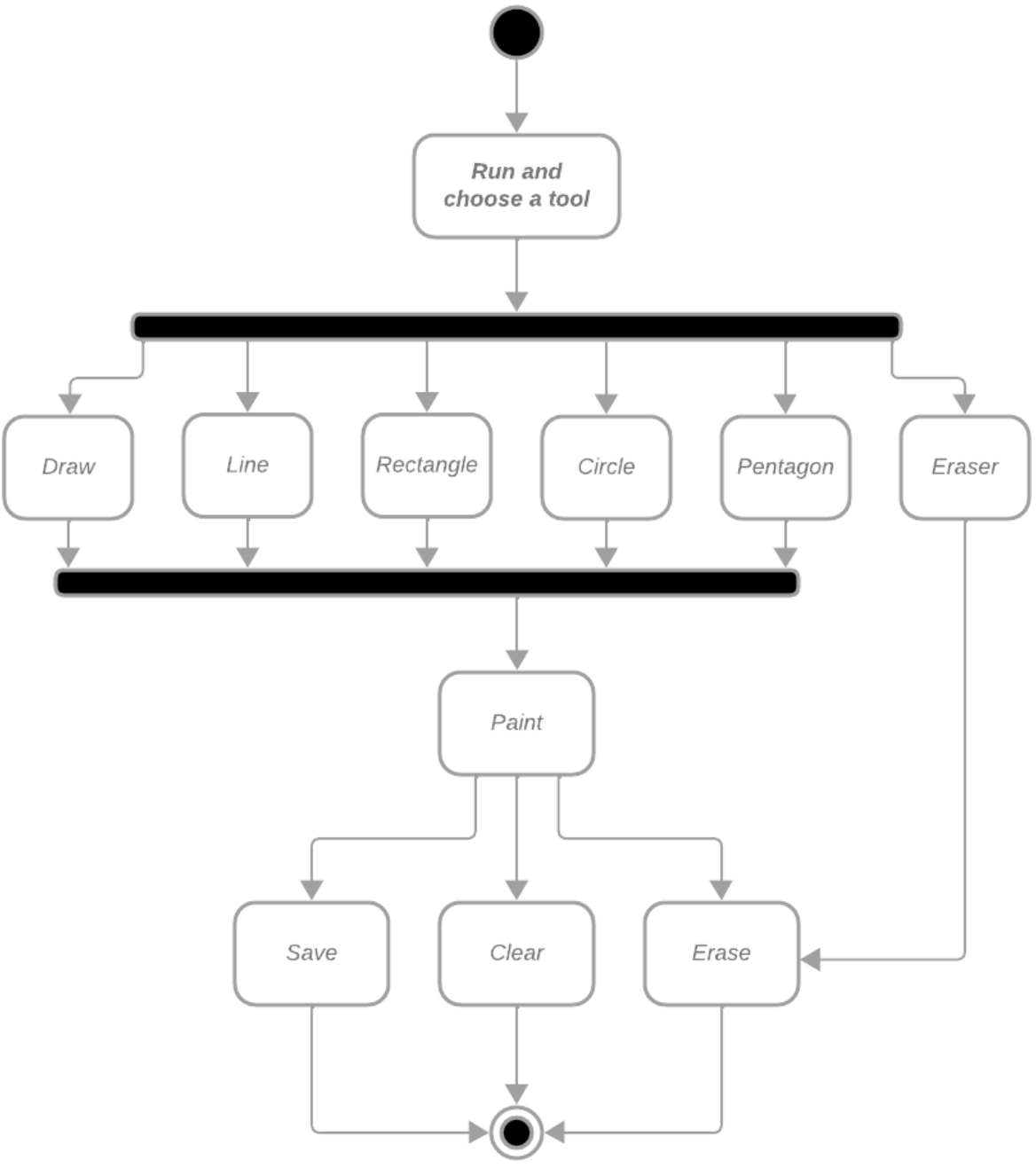

Fig-4.3.2: Activity diagram

IMPLEMENTATION
AND
RESULTS

5. IMPLEMENTATION AND RESULTS

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can consider being the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Recognizing the position of the writing hand and recognizing it through other gestures is an essential step in initializing aerial writing. Unlike traditional writing, when the pen moves down, and the pen moves up, writing in the air is not outlined as a writing sequence. Events. The system recognizes the position of a writing hand and distinguishes it from a non-writing hand by counting the number of raised fingers.

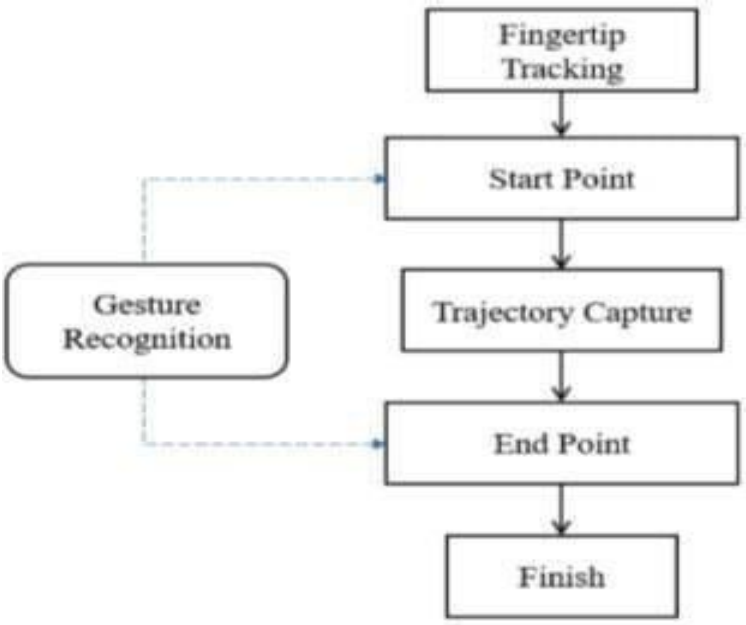

Fig 5.1: Implementation

Hand Region Segmentation

Once we have accurately captured the hand using the above technique, the segmentation of the hand area is done using a two-step approach, viz. The skin segmentation and the subtraction of the background and the final binary image of the hand are obtained as an aggregation of the two. The proposed algorithm works well in real-time and provides relatively accurate segmentation results. Although skin colors vary greatly from breed to breed, it has been observed that skin color has a small area between different skin types, while skin luminosity differs significantly.

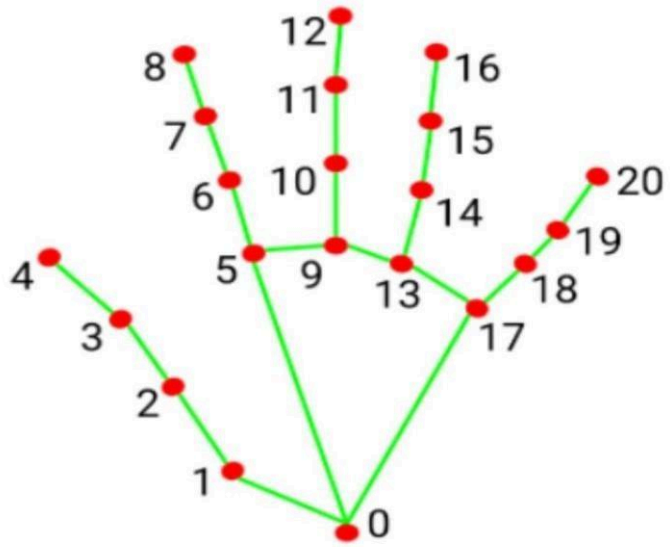

Fig 5.2: Fingertip detection

5.1 SOURCE CODE

```
import mediapipe as
mp import cv2
import numpy as np
import time
#
Constan
ts ml =
500
max_x, max_y = 250 + ml,
50 curr_tool = "Choose a
tool" time_init = True
rad = 40
var_inits =
False thick =
4
prevx, prevy =
0, 0 # Get tools
function def
getTool(x):
    if x < 40 +
        ml: return
        "Draw"
    elif x < 80 +
        ml: return
        "Line"
 elif x < 120 +
```

```
        ml: return
        "Circle"
elif x < 160 + ml:
        return
        "Rectangle"
```

```
 elif x < 200 + ml:
        return
        "Pentagon"
    else:
        return "Eraser"
def index_raised(yi,
    y9): if (y9 - yi) >
    40:
        return
    True
    return
    False
# Initialize MediaPipe Hands
hands = mp.solutions.hands
hand_landmark = hands.Hands(min_detection_confidence=0.6,
min_tracking_confidence=0.6, max_num_hands=1)
draw =
mp.solutions.drawing_utils #
Drawing tools
tools = cv2.imread("t4.png")
tools = tools.astype('uint8')
mask = np.ones((720, 1280), dtype=np.uint8) * 255
# Create a flag to indicate if the user wants to save the drawing
save_drawing = False
# Open the camera
cap = cv2.VideoCapture(0)
cap.set(3, 1280)
```

```
cap.set(4, 720)
```

```
# Create a flag to indicate if the user wants to clear the screen
clear_screen = False
while True:
    _, frm =
    cap.read() frm =
    cv2.flip(frm, 1)
    rgb = cv2.cvtColor(frm,
    cv2.COLOR_BGR2RGB) op =
    hand_landmark.process(rgb)
    if op.multi_hand_landmarks:
        for i in op.multi_hand_landmarks:
            draw.draw_landmarks(frm, i,
            hands.HAND_CONNECTIONS) x, y = int(i.landmark[8].x
            * 1280), int(i.landmark[8].y * 720)
            if x < max_x and y < max_y and x >
                ml: if time_init:
                    ctime =
                    time.time(
                    time_init =
                    False ptime =
                    time.time()
                cv2.circle(frm, (x, y), rad, (0, 255, 255), 2)
                rad -= 1
                if (ptime - ctime) >
                    0.8: curr_tool =
                    getTool(x)
                    print("Your current tool is set to:", curr_tool)
```

```
time_init = True

rad = 40
```

```
else:
                                                t
                                                i
                                                m
                                                e
                                                _
                                                i
                                                n
                                                i
                                                t
                                                =
                                                T
                                                r
                                                u
                                                e
                                                r
                                                a
                                                d
                                                =
                                                4
                                                0
                                            if curr_tool == "Draw":
                                                xi, yi =
                                                int(i.landmark[12].x *
                                                1280),
                                                int(i.landmark[12].y *
                                                720) y9 =
                                                int(i.landmark[9].y *
```

```
720)
if index_raised(yi, y9):
    cv2.lin
    e(mask
    ,
    (prevx,
    prevy),
    (x, y),
    0,
    thick)
    prevx,
    prevy
    = x, y
else:
    p
    r
    e
    v
    x
    =
    x
    p
    r
    e
    v
    y
    =
    y
elif curr_tool == "Line":
    xi, yi =
        int(i.landmark[12].
        x * 1280),
        int(i.landmark[12].
        y * 720) y9 =
        int(i.landmark[9].y
        * 720)
        i
            f
            i
            n
            d
            e
            x
            _
            r
            a
            i
            s
            e
            d
            (
            y
            i
            ,
            y
            9
```

```
)                                   ,
:                                   y
i                                   v
f                                   a
n                                   r
o                                   _
t                                   i
(                                   n
v                                   i
a                                   t
r                                   s
_                                   =
i                                   T
n                                   r
i                                   u
t                                   e
s                                   cv2.lin
)                                   e(frm,
:                                   (xii, yii),
    x                               (x, y),
    i                               (255, 0,
    i                               0), thick)
    ,                               else:
    y                               if var_inits:
    i                               cv2.
    i                               line(
    =                               mas
    x                               k,
```

```
(xii,
yii),
(x,
y),
0,
thic
k)
var_
inits
=
Fals
e
```

```
elif curr_tool == "Rectangle":
        xi, yi = int(i.landmark[12].x * 1280), int(i.landmark[12].y *
        720) y9 = int(i.landmark[9].y * 720)
        if index_raised(yi,
           y9): if not
           (var_inits):
              xii, yii = x, y
              var_inits =
              True
           cv2.rectangle(frm, (xii, yii), (x, y), (255, 0, 0),
        thick) else:
           if var_inits:
              cv2.rectangle(mask, (xii, yii), (x, y), 0, thick)
              var_inits = False
    elif curr_tool == "Circle":
        xi, yi = int(i.landmark[12].x * 1280), int(i.landmark[12].y *
        720) y9 = int(i.landmark[9].y * 720)
        if index_raised(yi,
           y9): if not
           (var_inits):
              xii, yii = x, y
              var_inits =
              True
           cv2.circle(frm, (xii, yii), int(((xii - x) ** 2 + (yii - y) ** 2) ** 0.5), (255, 0, 0),
        thick) else:
           if var_inits:
              cv2.circle(mask, (xii, yii), int(((xii - x) ** 2 + (yii - y) ** 2) **
```

```
0.5), (0, 255, 0), thick)
```

```
        var_inits = False
# Add this function to calculate the coordinates of a
pentagon def calculate_pentagon_coords(x1, y1, x2, y2):
    # Calculate the coordinates of the five points of the
    pentagon pentagon_coords = []
    angle = 360 / 5 # Angle between each point of the
    pentagon for i in range(5):
                x = int(x1 + (x2 - x1) * np.cos(np.deg2rad(i * angle)) - (y2 - y1) *
        np.sin(np.deg2rad(i * angle)))
                y = int(y1 + (x2 - x1) * np.sin(np.deg2rad(i * angle)) + (y2 - y1) *
        np.cos(np.deg2rad(i * angle)))
        pentagon_coords.append((x
    , y)) return pentagon_coords
# Inside the main loop, add a condition for the "Pentagon"
tool if curr_tool == "Pentagon":
    xi, yi = int(i.landmark[12].x * 1280), int(i.landmark[12].y *
    720) y9 = int(i.landmark[9].y * 720)
    if index_raised(yi,
        y9): if not
        (var_inits):
            xii, yii = x, y
            var_inits =
            True
        # Calculate the coordinates of the pentagon
        pentagon_coords = calculate_pentagon_coords(xii, yii,
        x, y)
```

```
# Draw the pentagon
                for p in range(len(pentagon_coords) - 1):
                    cv2.line(frm, pentagon_coords[p], pentagon_coords[p + 1], (255, 0, 0), thick)
                  cv2.line(frm, pentagon_coords[-1], pentagon_coords[0], (255, 0, 0), thick)
          else:
                if var_inits:
                    # Calculate the coordinates of the pentagon
                    pentagon_coords = calculate_pentagon_coords(xii, yii,
                    x, y) # Draw the pentagon on the mask
                    for p in range(len(pentagon_coords) - 1):
                        cv2.line(mask, pentagon_coords[p], pentagon_coords[p + 1], 0,
                    thick) cv2.line(mask, pentagon_coords[-1], pentagon_coords[0], 0,
                    thick) var_inits = False
           elif curr_tool == "Eraser":
              xi, yi = int(i.landmark[12].x * 1280), int(i.landmark[12].y *
              720) y9 = int(i.landmark[9].y * 720)
              if index_raised(yi, y9):
                  cv2.circle(frm, (x, y), 30, (0, 0, 0), -1)
                  cv2.circle(mask, (x, y), 30, 255, -1)
    mask = mask.astype('uint8') # Ensure mask is of the correct data
    type op = cv2.bitwise_and(frm, frm, mask=mask)
    frm[:, :, 1] = op[:, :, 1]
    frm[:, :, 2] = op[:, :, 2]
    frm[:max_y, ml:max_x] = cv2.addWeighted(tools, 0.7, frm[:max_y, ml:max_x], 0.3, 0)
```

```
cv2.putText(frm, curr_tool, (270 + ml, 30), cv2.FONT_HERSHEY_TRIPLEX, 1, (0, 0, 0),
  2) if clear_screen:
      mask = np.ones((720, 1280), dtype=np.uint8) * 255
       clear_screen = False
   cv2.imshow("Painting interface",
   frm) key = cv2.waitKey(1)
   if key == 27: # 'Esc' key to
      exit break
   elif key == ord('s'): # 's' key to save the drawing
      filename = f'drawing_{int(time.time())}.png'
      cv2.imwrite(filename, mask)
      print(f"Drawing saved as {filename}")
   elif key == ord('S'): # 'S' key to save the drawing
      filename = f'drawing_{int(time.time())}.png'
      cv2.imwrite(filename, mask)
      print(f"Drawing saved as {filename}")
   elif key == ord('c'): # 'c' key to clear the screen
      clear_screen = True
   elif key == ord('C'): # 'c' key to clear the screen
      clear_screen = True
cv2.destroyAllWindows()
cap.release()
```

5.2 OUTPUT SCREENS

Fig-5.2.1: Painting interface

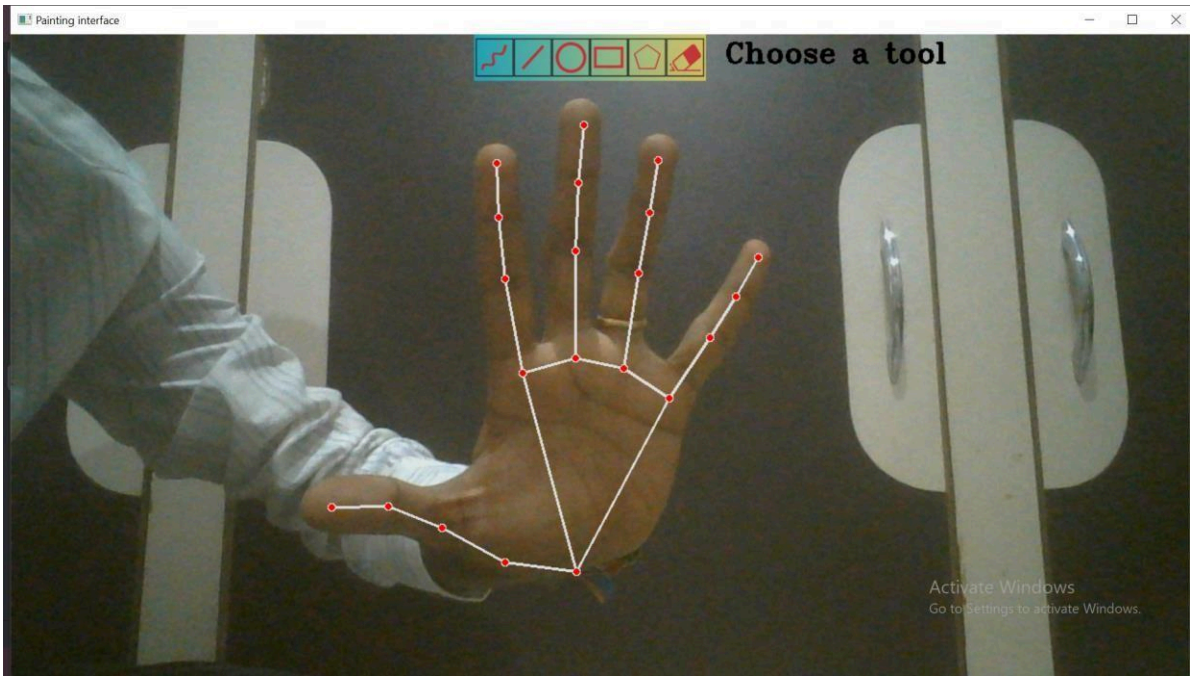

Fig-5.2.2: Hand detection

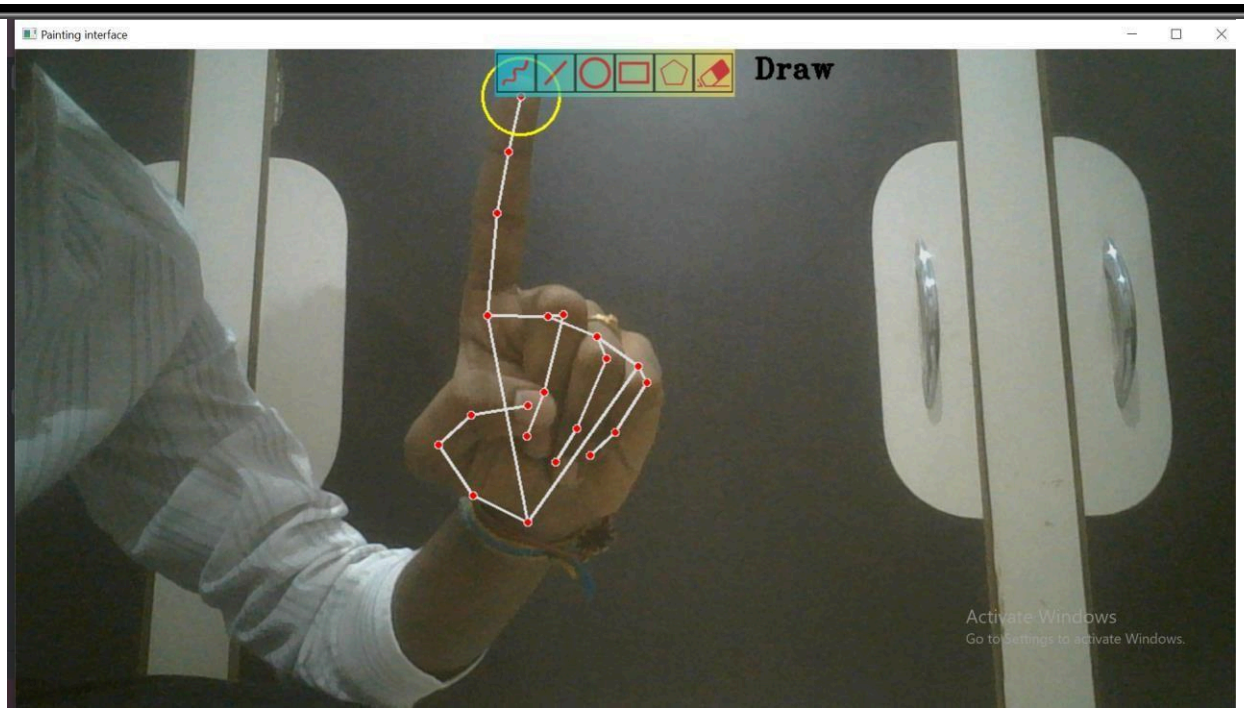

Fig-5.2.3: Selecting Draw tool

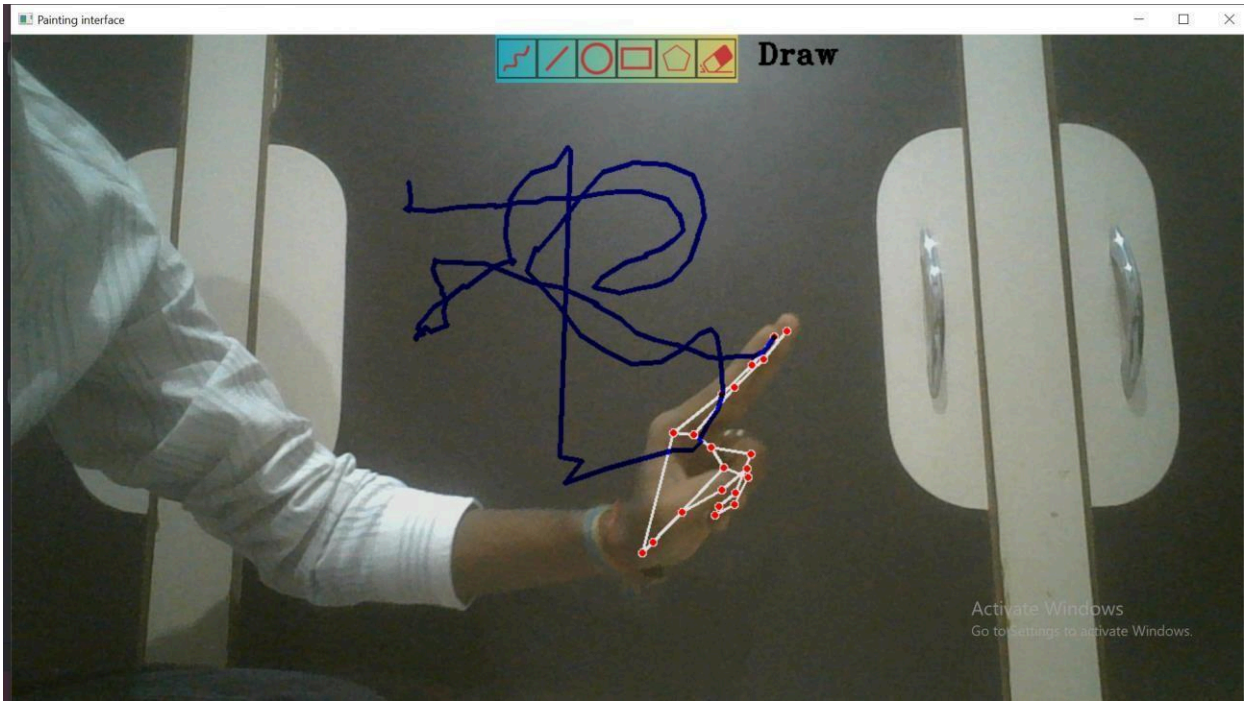

Fig-5.2.4: Free hand Drawing

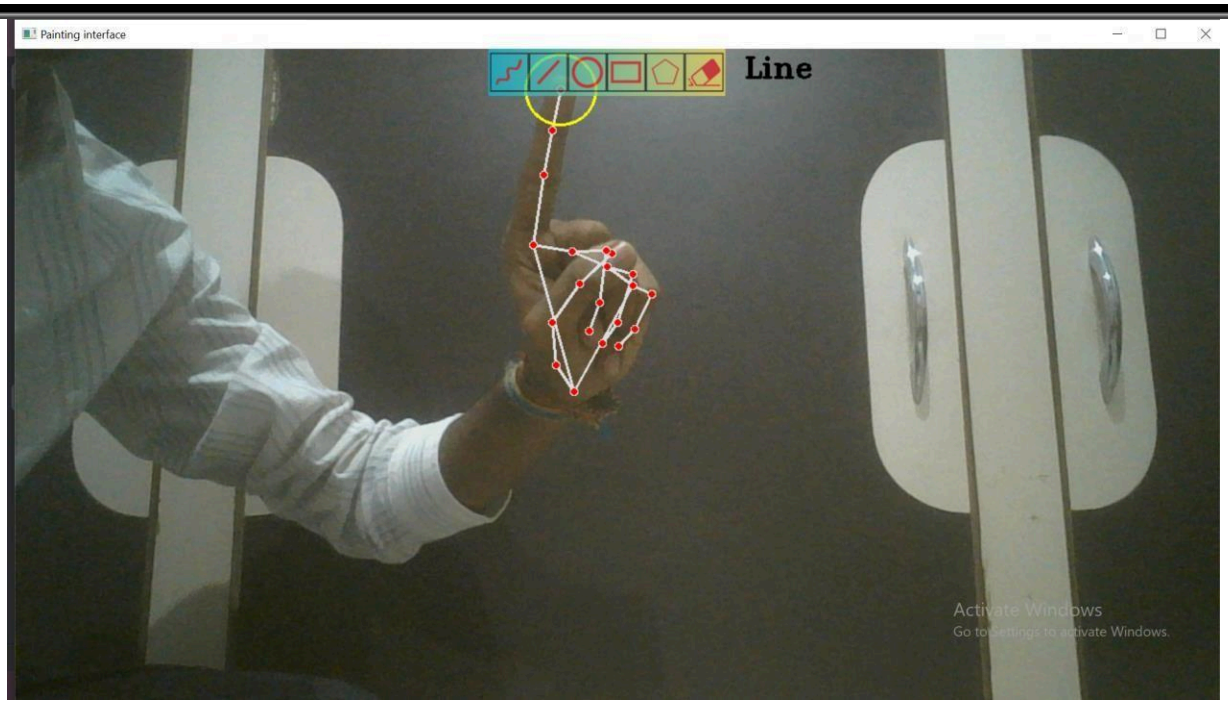

Fig-5.2.5: Selecting Line tool

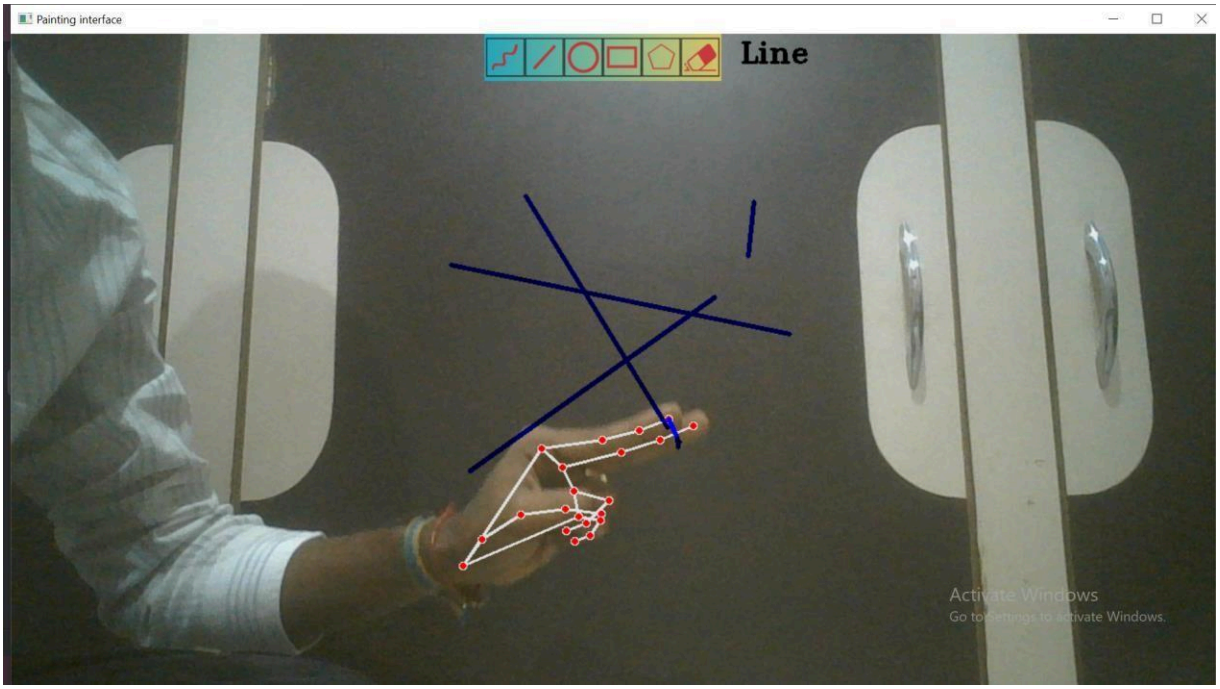

Fig-5.2.6: Drawing Line

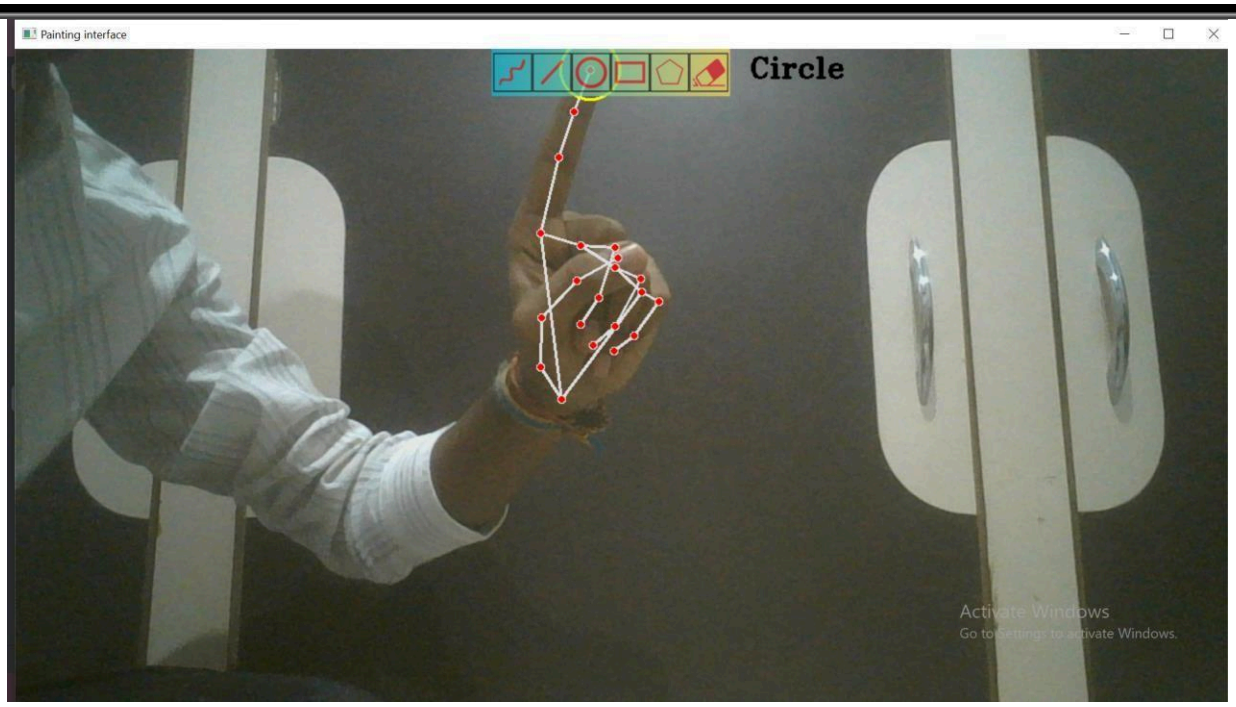

Fig-5.2.7: Selecting Circle tool

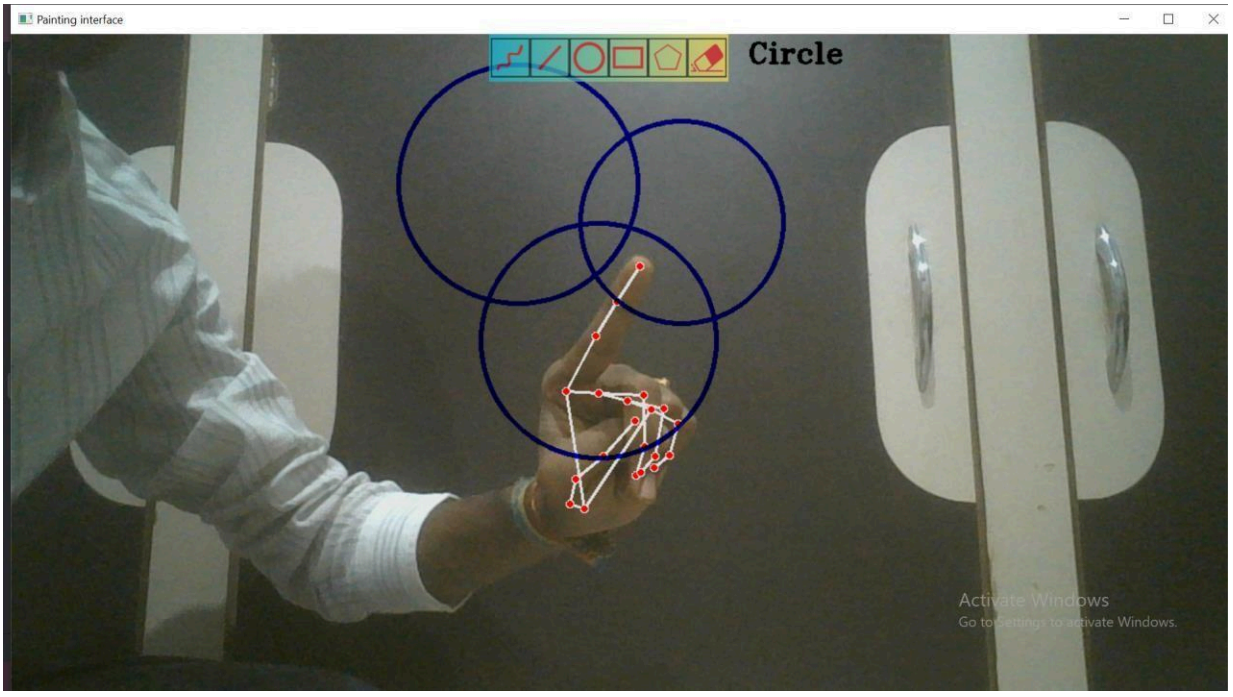

Fig-5.2.8: Drawing Circle

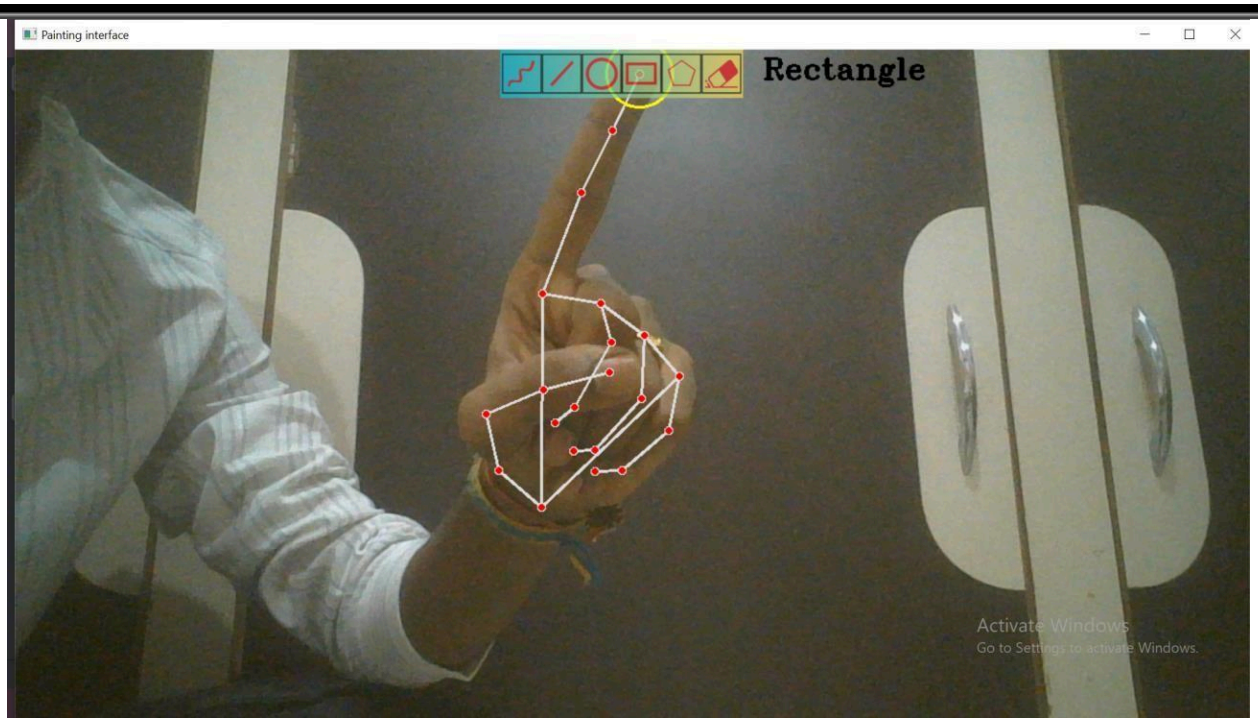

Fig-5.2.9: Selecting Rectangle tool

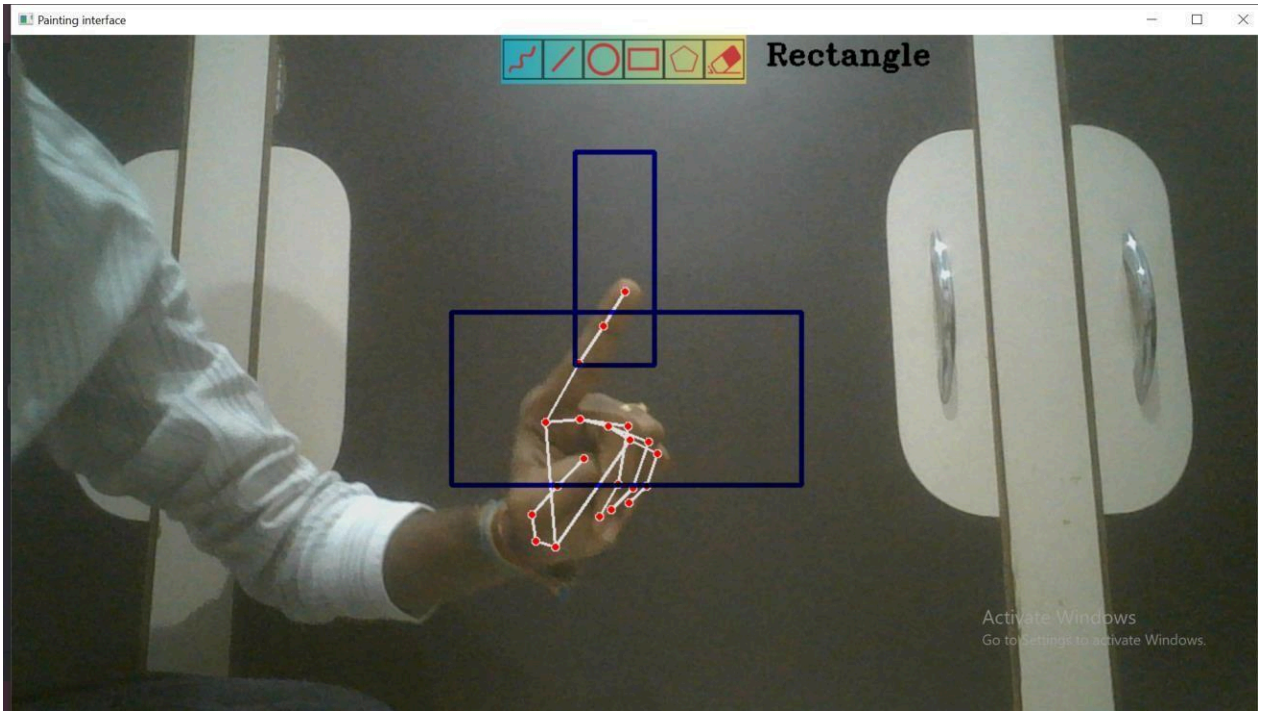

Fig-5.2.10: Drawing Rectangle

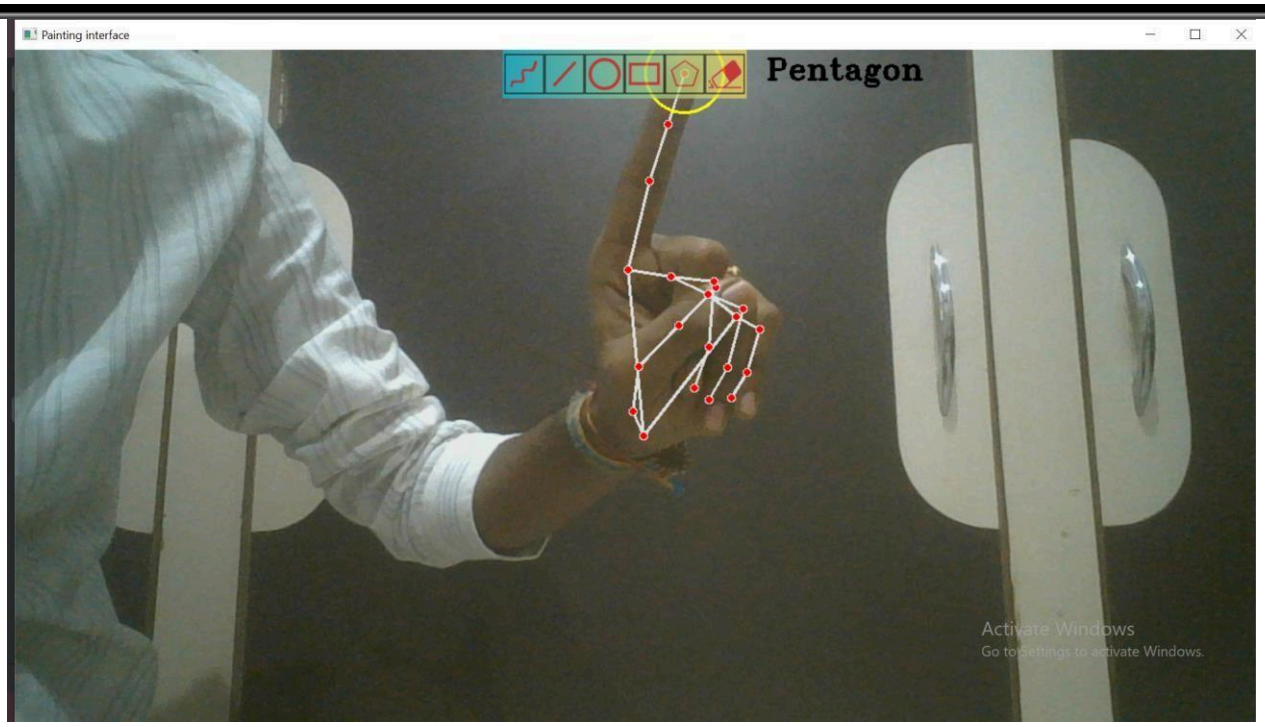

Fig-5.2.11: Selecting Pentagon tool

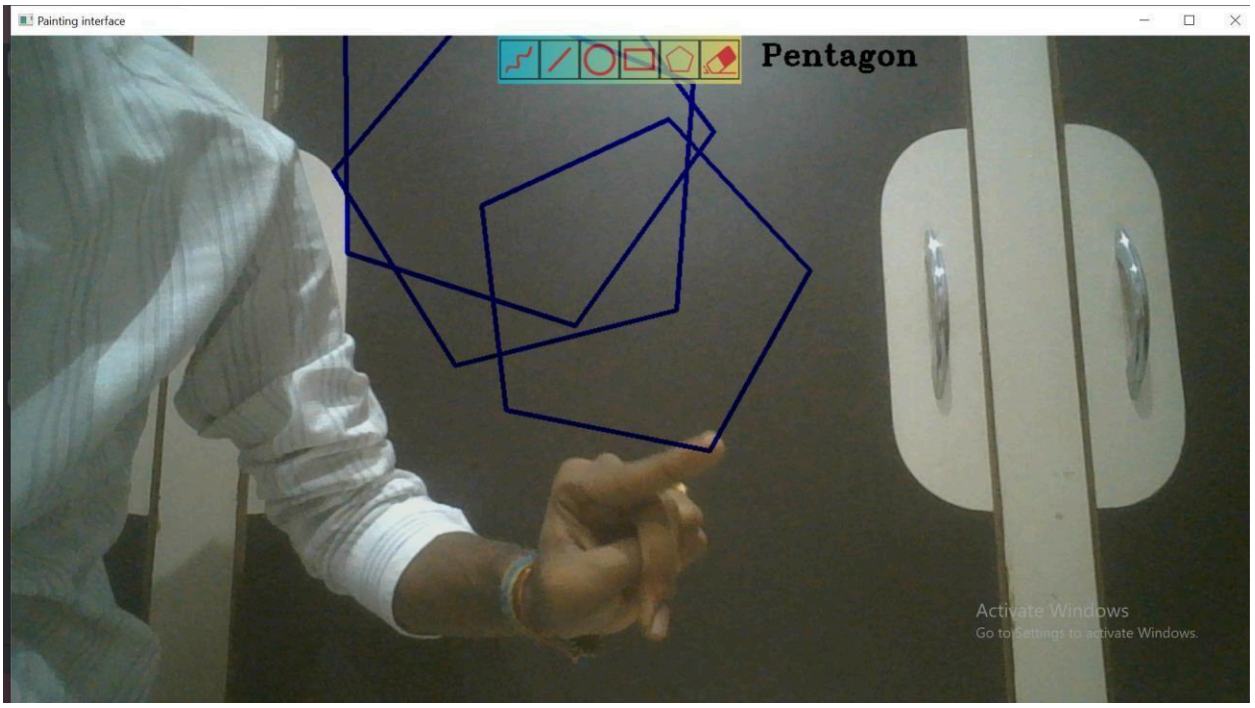

Fig-5.2.12: Drawing Pentagon

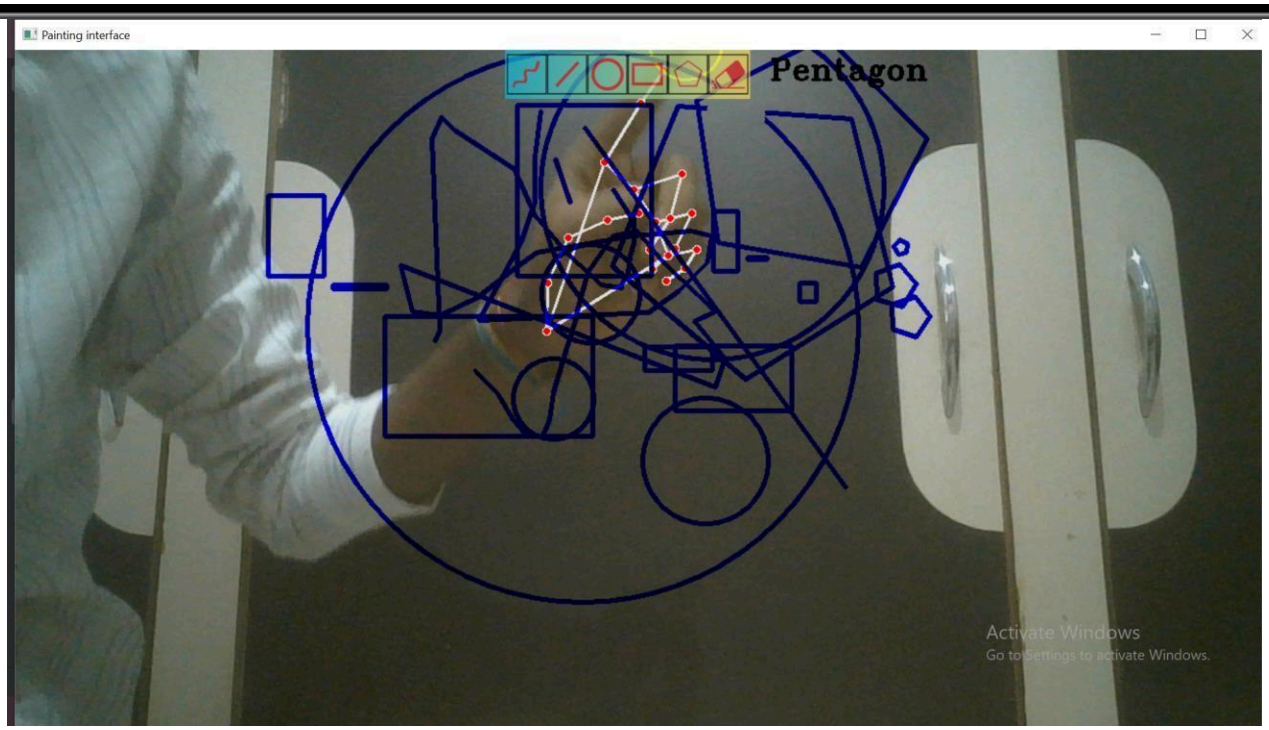

Fig-5.2.13: Performed actions

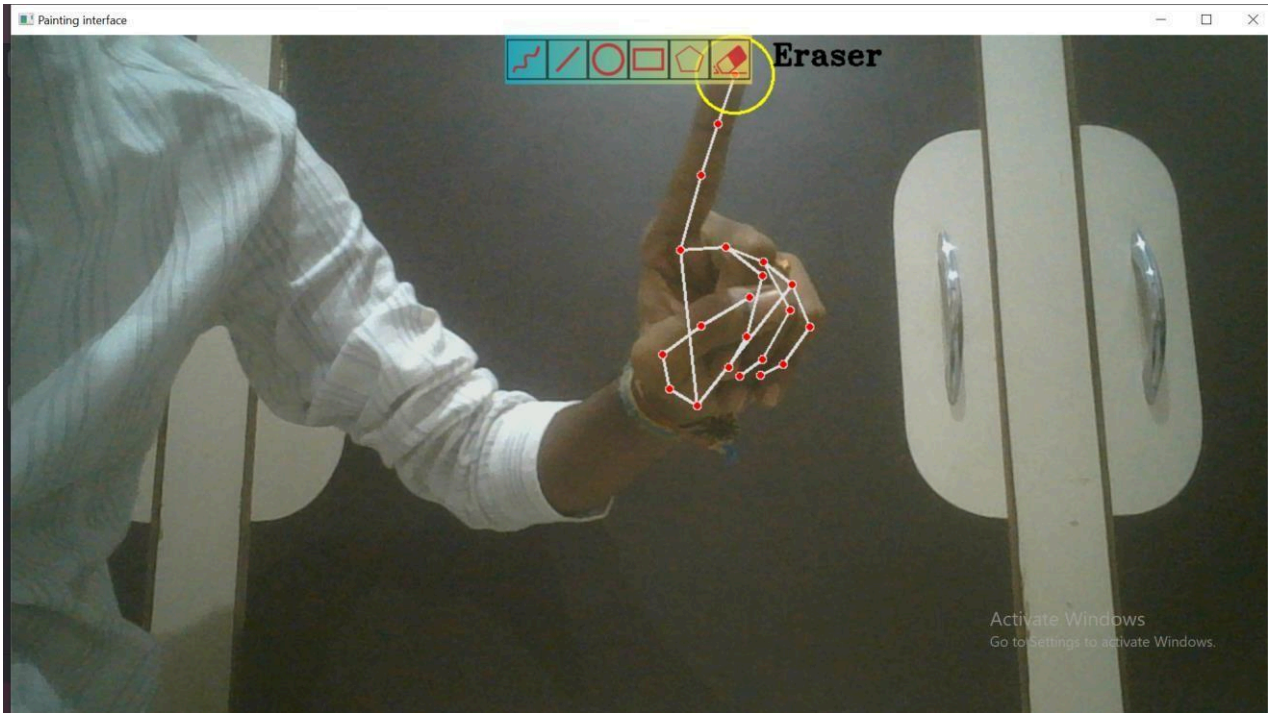

Fig-5.2.14: Selecting Eraser tool

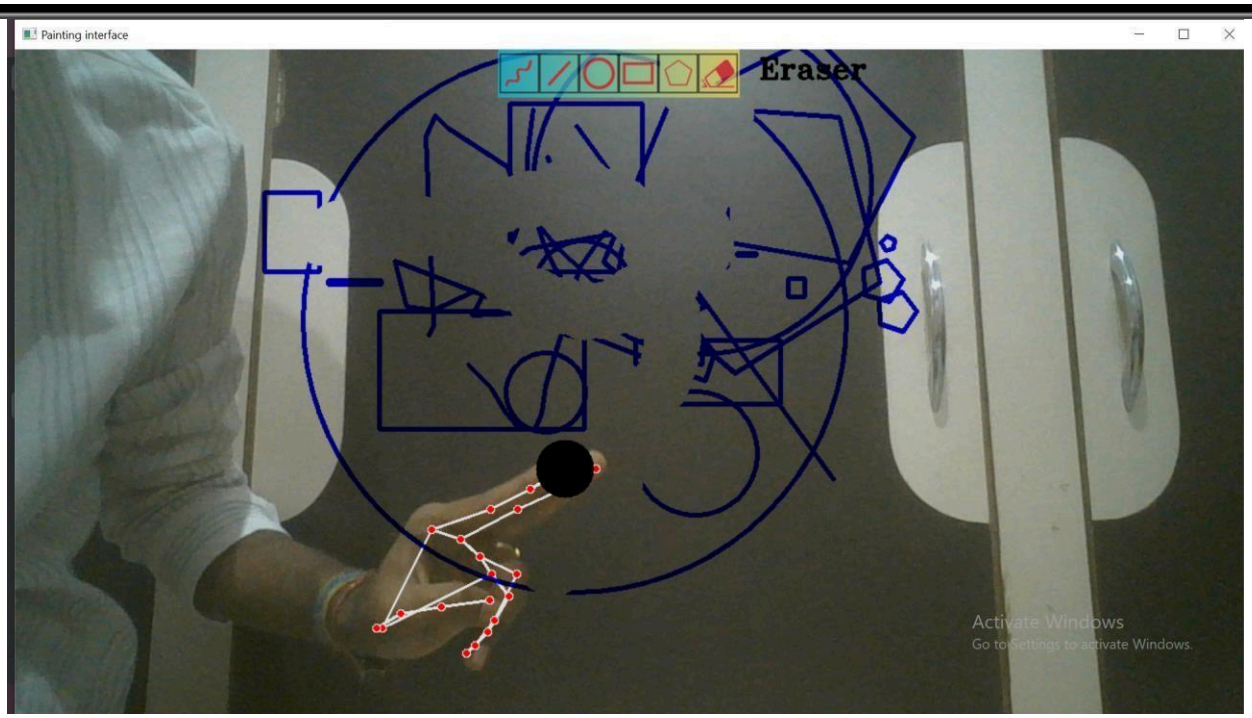

Fig-5.2.15: Erasing the screen

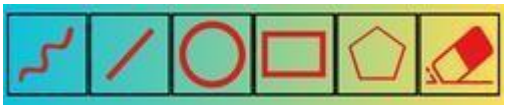

Fig-5.2.16: Tool bar

Fig-5.2.17: Saved Image 1

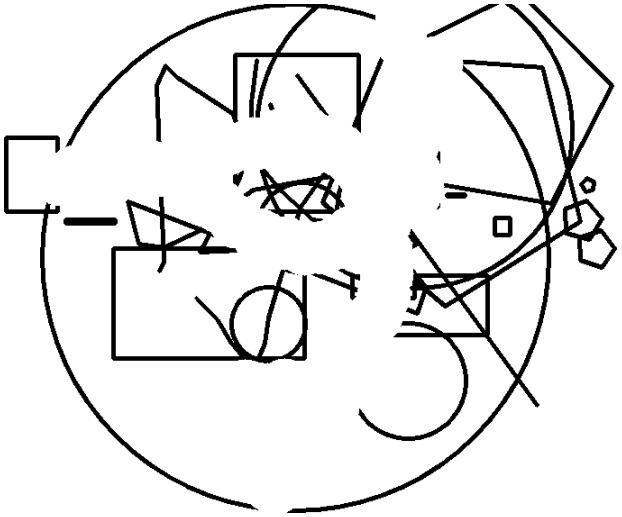

Fig-5.2.18: Saved Image 2

TESTING AND VALIDATION

6. TESTING AND VALIDATION

Tool Selection Testing

- Verify that each tool (e.g., brush, line, circle, rectangle, pentagon, eraser) can be selected and used accurately.
- Ensure that tool selection is responsive and intuitive.

Tool Switching Testing

- Verify that users can switch between tools seamlessly, either through hand gestures or on- screen menu options.

Drawing Interaction Testing

- Test each drawing tool to ensure they function as intended.

Canvas Management Testing

- Test saving drawings to ensure they are saved correctly as image files (e.g., PNG).

  Verify that the "Clear Screen" option erases the entire canvas.

User Interface Testing

- Confirm that the user interface displays the selected tool, and other relevant information clearly.
- Ensure that the interface elements are responsive and update in real-time.

CONCLUSION
AND
FUTURE ENHANCEMENTS

7. CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

This project could put conventional writing techniques to the test. Removes the need to always have a cell phone with you in order to take notes, making it convenient to do so when on the go. The application can be used by people who have trouble using the keyboard. This program is getting more functional, and soon it will be able to control IoT devices. This software will be an excellent tool for smart clothes, enabling users to interact with the digital world more effectively. Programs that write in the wind should only follow their master's instructions and not be led away by outside influences.

The following discovery algorithms, like YOLO v3, can increase the speed and accuracy of fingerprint identification. The effectiveness of writing in the air will increase as artificial intelligence advances.

7.2 FUTURE ENHANCEMENTS

Import and Edit Images: Enable users to import external images into the drawing software for editing, manipulation, and incorporation into their artwork.

Cloud Storage and Collaboration: Implement cloud storage integration to allow users to save their artwork online and access it from multiple devices.

VR Headset Support: Enable compatibility with virtual reality (VR) headsets to provide an immersive drawing experience in a virtual environment. Users can create artwork in 3D space and interact with their creations using VR controllers.

AI-Powered Art Assistance: Incorporate artificial intelligence (AI) algorithms to assist users in various aspects of their artwork, such as suggesting brush strokes, color palettes, composition, or even generating artwork based on user input.

REFERENCES

8. REFERENCES

[1] Y. Huang, X. Liu, X. Zhang, and L. Jin, "A Pointing Gesture Based Egocentric Interaction System: Dataset, Approach, and Application," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV. pp. 370-377, 2016.

[2] Saira Beg, M. Fahad Khan and Faisal Baig, "Text Writing in Air," Journal of Information Display Volume 14, Issue 4, 2013

[3] Yuan-Hsiang Chang, Chen-Ming Chang, "Automatic Hand-Pose Trajectory Tracking System Using Video Sequences", INTECH, pp. 132- 152, Croatia, 2010