A

Project Report On

#### HAND GESTURE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS

Submitted in partial fulfillment of the requirements for the award of Degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

By

**M.KARISMA (207R1A66F7)**

**P.SAI TEJA (207R1A66G6)**

**A.PRAVEEN (207R1A66C4)**

Under the Guidance of

**Mrs. G. Parvathi Devi**

Assistant Professor of CSE (AI & ML)

##### **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

**CMR TECHNICAL CAMPUS UGC AUTONOMOUS**

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401. **2020-2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)**



#### CERTIFICATE

This is to certify that the project entitled **“ HAND GESTURE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS ”** being submitted by **M.KARISMA (207R1A66F7), P.SAI TEJA (207R1A66G6) & A.PRAVEEN (207R1A66C4)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (AI&ML) to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

**Mrs. G. Parvathi Devi Dr. S Rao Chintalapudi**

Assistant Professor of CSE(AI&ML) HOD CSE(AI&ML)

INTERNAL GUIDE

**EXTERNAL EXAMINER**

**Submitted for viva voice Examination held on** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGEMENT**

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mrs. G. Parvathi Devi,** Assistant Professor of CSE(AI&ML) for her exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by her shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. G. Vinoda Reddy, Dr. K. Mahesh, Mrs. B. Mamatha & Mr. N. Sateesh** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. S Rao Chintalapudi,** Head, Department of Computer Science and Engineering (AI&ML) for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

###### **M.KARISMA (207R1A66F7)**

**P.SAI TEJA (207R1A66G6)**

**A.PRAVEEN (207R1A66C4)**

**ABSTRACT**

Hand Gesture Recognition (HGR) targets on interpreting the sign language into text or speech, so as to facilitate the communication between deaf-mute people and ordinary people. This task has broad social impact, but is still very challenging due to the complexity and large variations in hand actions. Existing methods for HGR use hand-crafted features to describe sign language motion and build classification models based on those features. However, it is difficult to design reliable features to adapt to the large variations of hand gestures.

To approach this problem, we proposed Convolutional Neural Networks (CNN) which extracts discriminative spatial-temporal features from raw video stream automatically without any prior knowledge, avoiding designing features. We validate the proposed model on a real dataset collected with Microsoft Kinect and demonstrate its effectiveness over the traditional approaches based on hand-crafted features. For gesture recognition, users can upload test images or use a webcam to capture real-time video frames. The system processes the input images by converting them to grayscale, applying Gaussian blur, and thresholding to extract hand gestures. The processed images are resized and fed into the trained CNN model for classification. The predicted gesture is displayed along with the input image. The project aims to provide a user-friendly interface for hand gesture recognition, facilitating applications such as sign language translation, human-computer interaction, and virtual reality control. Future enhancements could include improving model accuracy, optimizing computational efficiency, and expanding gesture recognition capabilities to support a broader range of gestures and applications.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| FIGURE NO | FIGURE NAME | PAGE NO |
| 3.1 | Project Architecture for  Hand Gesture Recognition Using  Convolutional Neural Networks | 11 |
| 3.2 | Use Case Diagram for  Hand Gesture Recognition Using  Convolutional Neural Networks | 13 |
| 3.3 | Class Diagram for  Hand Gesture Recognition Using  Convolutional Neural Networks | 15 |
| 3.4 | Sequence diagram for  Hand Gesture Recognition Using  Convolutional Neural Networks | 17 |
| 4.3 | Confusion Matrix | 25 |

|  |  |  |
| --- | --- | --- |
| 5.1 | Main Page | 35 |
| 1_7YsXhDGKc9F2jKZvvm7FOA5.2 | Dataset Uploading Page | 35 |
| 5.3 | Testing Accuracy Page | 36 |
| 5.4 | Selecting Image Page | 36 |
| 5.5 | Displays Result Page | 37 |
| 5.6 | Selecting Video Page | 37 |
| 5.7 | Displays Result Page | 38 |
| 5.8 | Displays Result Page | 38 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| TABLE NO. | TABLE NAME | PAGE NO. |

**6.3** TEST CASES 42



**TABLE OF CONTENTS**

**ABSTRACT** i

LIST OF FIGURES ii

LIST OF TABLES iv

1. [INTRODUCTION](#_bookmark0) 1
   1. [PROJECT SCOPE 2](#_bookmark1)
   2. [PROJECT PURPOSE 2](#_bookmark2)
   3. [PROJECT FEATURES 3](#_bookmark3)
2. [SYSTEM ANALYSIS 4](#_bookmark4)
   1. [PROBLEM DEFINITION 5](#_bookmark5)
   2. [EXISTING SYSTEM 5](#_bookmark6)
      1. LIMITATIONS OF THE EXISTING SYSTEM 6
   3. [PROPOSED SYSTEM 7](#_bookmark7)
      1. ADVANTAGES OF PROPOSED SYSTEM 8
   4. [HARDWARE & SOFTWARE REQUIREMENTS 9](#_bookmark11)
      1. [HARDWARE REQUIREMENTS 9](#_bookmark12)
      2. [SOFTWARE REQUIREMENTS 9](#_bookmark13)
3. [ARCHITECTURE 1](#_bookmark14)0
   1. [PROJECT ARCHITECTURE 1](#_bookmark15)1
   2. [USE CASE DIAGRAM 1](#_bookmark17)3
   3. [CLASS DIAGRAM 1](#_bookmark18)5
   4. [SEQUENCE DIAGRAM 1](#_bookmark19)7
4. [IMPLEMENTATION 1](#_bookmark21)9
   1. CONVOLUTIONAL NEURAL NETWORK 20
   2. DATASET DESCRIPTION 21
   3. PERFORMANCE METRICS 22
   4. SAMPLE CODE 26
   5. [RESULT ANALYSIS 33](#_bookmark22)
5. [SCREENSHOTS 3](#_TOC_250001)4
6. [TESTING 3](#_bookmark23)9
   1. [INTRODUCTION TO TESTING 4](#_bookmark24)0
   2. [TYPES OF TESTING 4](#_bookmark25)0
      1. [UNIT TESTING 4](#_bookmark26)0
      2. [INTEGRATION TESTING 4](#_bookmark27)0
      3. [FUNCTIONAL TESTING 4](#_bookmark28)0
   3. [TEST CASES 4](#_bookmark29)1
7. [CONCLUSION & FUTURE SCOPE 4](#_TOC_250000)3
   1. [PROJECT CONCLUSION](#_bookmark31) 44
   2. [FUTURE SCOPE](#_bookmark32) 44

BIBLIOGRAPHY 46

[REFERENCES 47](#_bookmark33)

[GITHUB LINK 47](#_bookmark34)

1. **INTRODUCTION**

#### INTRODUCTION

* 1. **PROJECT SCOPE**

The aim of this project is to develop a web platform that can recognize hand gestures using datasets and CNN. The user will search for various hand gestures by selecting different images. The scope of a Hand Gesture Recognition project using Convolutional Neural Networks (CNNs) encompasses data collection, model architecture design, training, and evaluation.

Hand Gesture Recognition using Convolutional Neural Networks (CNNs) involves developing a system that can accurately identify and interpret hand gestures from images or video streams. The project encompasses various stages, including data collection, preprocessing, model selection, training, evaluation, deployment, and user interface design. It aims to create a robust and user-friendly solution applicable in diverse fields such as sign language recognition, human-computer interaction, virtual reality, and robotics. Ethical considerations regarding data privacy, bias, and regulatory compliance are integral throughout the project. Additionally, continual testing, iteration, and documentation ensure the system's effectiveness, reliability, and transparency. Future enhancements may involve exploring multi-modal recognition, real-time performance optimization, and expanding gesture vocabulary to further advance the capabilities and applications of the system.

* 1. **PROJECT PURPOSE**

The purpose of a Hand Gesture Recognition using Convolutional Neural Networks (CNNs) is to develop a system capable of accurately interpreting hand gestures captured through images or videos. By leveraging CNNs, the project aims to automatically extract meaningful features from hand gesture data, enabling applications such as human-computer interaction, sign language recognition, virtual reality control, and robotics. This technology enhances user experiences by providing intuitive interfaces and facilitating communication for individuals with disabilities. Moreover, it opens avenues for innovative interaction methods in diverse fields, ultimately improving efficiency, accessibility, and inclusivity in human-machine interaction.

Hand Gesture Recognition using Convolutional Neural Networks project is to develop an automated system capable of accurately detecting and interpreting hand gestures from image or video inputs. By leveraging deep learning techniques, particularly CNNs, the project aims to enable seamless interaction between humans and machines, with applications spanning various domains such as sign language interpretation, gesture-based control interfaces, augmented reality, and robotics. The project seeks to enhance accessibility, improve user experience, and facilitate intuitive communication modalities, ultimately advancing the capabilities of human-computer interaction and expanding the potential for innovative technological solutions in both commercial and assistive contexts.

* 1. **PROJECT FEATURES**

A Hand-gesture recognition project utilizing Convolutional Neural Networks (CNNs) involves several key components to accurately interpret and respond to hand movements. Beginning with the collection of a diverse dataset of hand gesture images, preprocessing techniques such as resizing and normalization are applied. The core of the project lies in designing an effective CNN architecture, typically comprising convolutional and pooling layers for feature extraction, followed by fully connected layers for classification. Training the model involves optimizing parameters using algorithms like stochastic gradient descent. Evaluation on a separate dataset ensures the model's accuracy and performance. Real-time processing and user interface development are essential for practical application, allowing users to interact through hand gestures seamlessly. Robustness measures handle variations in lighting and occlusions, while a feedback mechanism provides users with confirmation of gesture recognition. Security and privacy considerations are also paramount. Overall, by integrating these features, a hand gesture recognition system can accurately interpret gestures for diverse applications with reliability and usability.

## **2.SYSTEM ANALYSIS**

##### **SYSTEM ANALYSIS**

**SYSTEM ANALYSIS**

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

* 1. **PROBLEM DEFINITION**

Hand Gesture Recognition Using Convolutional Neural Networks project refers to the development and implementation of a system capable of identifying and interpreting hand gestures through the utilization of CNNs. This project involves collecting a dataset of hand gesture images, preprocessing them, and designing a CNN architecture suitable for feature extraction and classification. The CNN is trained on the dataset using optimization algorithms to learn patterns and associations between hand gestures and their corresponding labels. Once trained, the model can accurately recognize hand gestures in real-time or offline scenarios. Applications of such projects span various domains, including human-computer interaction, sign language recognition, virtual reality control, and gesture-based command interfaces.

* 1. **EXISTING SYSTEM**

Hand gesture recognition systems can be developed without utilizing Convolutional Neural Networks (CNNs) by employing alternative machine learning and computer vision techniques. Such systems typically involve extracting relevant features from hand images, such as histograms of oriented gradients (HOG) or keypoints using methods like SIFT or SURF, followed by dimensionality reduction using techniques like PCA or t-SNE. Machine learning classifiers such as Support Vector Machines (SVM), Random Forests, or k-Nearest Neighbors (k-NN) are then trained on these features to recognize patterns corresponding to different hand gestures.

Real-time processing can be achieved by integrating hand detection and tracking

algorithms with the trained classifier, allowing for gesture recognition in real-time applications. These systems can find applications in various domains, including virtual reality, smart home control, and gaming, offering users intuitive and interactive ways to interact with digital interfaces without the need for CNNs.

While hand gesture recognition systems employing traditional machine learning and computer vision techniques offer valuable advantages, they also present notable drawbacks. One significant limitation lies in their susceptibility to variability across lighting conditions, backgrounds, and hand orientations, potentially compromising accuracy and reliability. Moreover, the manual feature engineering required for feature extraction can be time-intensive and may not fully capture the complexity of diverse gesture patterns, affecting the system's ability to recognize gestures accurately. Additionally, these systems heavily rely on hand-crafted features whose quality and relevance significantly impact performance; inadequacies in feature representation can lead to misclassification or erroneous recognition of gestures, especially in real-world settings with noise or occlusions. Furthermore, their limited adaptability to new gestures or variations not present in the training data poses challenges, often necessitating extensive retraining for system updates. Despite these challenges, traditional hand gesture recognition systems remain relevant in scenarios prioritizing interpretability, simplicity, or computational efficiency over absolute performance. However, ongoing advancements in deep learning techniques may gradually alleviate some of these limitations, potentially reshaping the landscape of gesture recognition systems in the future.

* + 1. **LIMITATIONS OF THE EXISTING SYSTEM**

Following are the disadvantages of existing system:

* Dependency on Hand-Crafted Features
* Susceptibility to Noise and Variability
* Inability to handle missing values effectively.
* Less Efficient in Complex Environments
* Lack of accurate probabilistic outputs.
* Limited Feature Representation
  1. **PROPOSED SYSTEM**

The proposed system for Hand Gesture Recognition Using Convolutional Neural Networks (CNNs) seeks to develop a highly accurate and efficient solution for real-time recognition and interpretation of hand gestures. This system involves collecting a diverse dataset of hand gesture images and preprocessing them to enhance quality and diversity. A custom CNN architecture will be designed and trained on this dataset, optimized to effectively extract features and classify gestures. Real-time processing capabilities will be integrated into the system, allowing for seamless interaction with live camera feeds or sensor inputs. The system will also include a user interface for intuitive interaction, with gestures mapped to specific actions or commands. Rigorous evaluation and testing will ensure the system's accuracy, robustness, and usability across various scenarios. Ultimately, the proposed system aims to provide a versatile and scalable solution for hand gesture recognition, applicable in diverse domains such as human-computer interaction, virtual reality, and robotics.

The proposed system for Hand Gesture Recognition Using Convolutional Neural Networks (CNNs) is designed to provide an accurate and efficient solution for real-time interpretation of hand gestures. It begins by assembling a diverse dataset of hand gesture images, ensuring that it encompasses a wide range of hand shapes, poses, and environmental conditions. These images undergo preprocessing stages aimed at enhancing their quality and diversity, which is essential for training a robust CNN model. A custom CNN architecture will then be meticulously crafted and trained on this dataset, leveraging the network's ability to automatically learn pertinent features directly from raw pixel data. Special emphasis will be placed on optimizing the architecture to efficiently extract discriminative features and classify gestures with a high degree of accuracy.

Real-time processing capabilities will be seamlessly integrated into the system, enabling it to interact with live camera feeds or sensor inputs in a responsive manner—a crucial requirement for applications necessitating immediate feedback. Moreover, the system will boast a user-friendly interface, carefully designed to facilitate intuitive interaction. Recognized gestures will be mapped to specific actions or commands, enhancing user experience and making the system accessible to a broad audience. Rigorous evaluation and testing procedures will be conducted to validate the system's accuracy, robustness, and usability across various scenarios, ensuring its reliability in real-world applications.

Ultimately, the proposed system aims to offer a versatile and scalable solution for hand gesture recognition, applicable across a spectrum of domains ranging from human-computer interaction to virtual reality and robotics. By providing a precise and intuitive means of interacting with digital interfaces and controlling devices, the system endeavors to enhance user experience and foster seamless integration of gesture-based interaction into diverse technological ecosystems.

**2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM**

* **High Accuracy:** Convolutional Neural Networks (CNNs) have demonstrated excellent performance in image recognition tasks, making them well-suited for accurately recognizing and classifying hand gestures.
* **Adaptability:** CNN architectures can be tailored to accommodate various hand gestures and environmental factors, ensuring adaptability to different users and settings.
* **Continuous Improvement:** As more data is collected and the model is further trained, the system can continuously improve its accuracy and performance, ensuring ongoing optimization and refinement.
* **Versatility:** The system's versatility enables its application across diverse domains, including human-computer interaction, virtual reality, gaming, robotics, and healthcare, among others.
* **Efficiency:** The optimized CNN architecture and real-time processing capabilities contribute to the system's efficiency, minimizing latency and computational resources required for gesture recognition.
* **User-Friendly Interface:** The integration of a user interface simplifies interaction, allowing users to intuitively control devices or applications through hand gestures without the need for additional peripherals or complex setups.**2.4 HARDWARE & SOFTWARE REQUIREMENTS**

###### **HARDWARE REQUIREMENTS:**

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

* PROCESSOR : i5 or above
* RAM : 4GB (min)
* HARD DISK : 20 GB
* KEYBOARD : Standard Windows Keyboard
* MOUSE : Two or Three Button Mouse
* MONITOR : SVGA

##### **SOFTWARE REQUIREMENTS:**

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

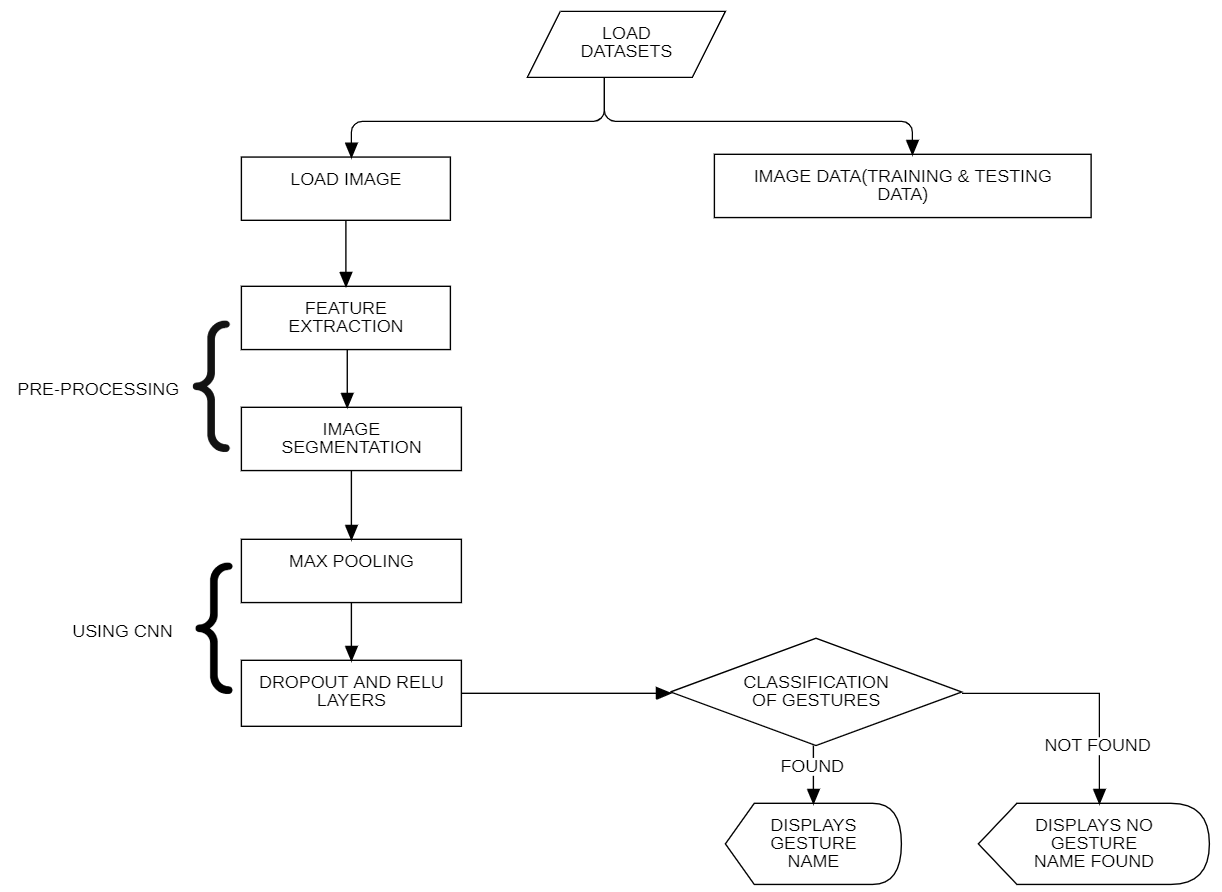
* OPERATING SYSTEM : Windows 10
* CODE LANGUAGE : Python
* LIBRARIES : cv2,tkinter,numpy,pickle,keras,os
* FRONT-END : Python
* BACK-END : Django-ORM
* DESIGNING : HTML, CSS, JavaScript
* DATABASE : MySQL (WAMP Server).

**3. ARCHITECTURE**

##### **3.ARCHITECTURE**

##### **PROJECT ARCHITECTURE**

This project architecture shows the procedure followed for classification, starting from input to final prediction.



NO

OUTPUT

Figure 3.1: Project Architecture of Hand Gesture Recognition

Using Convolutional Neural Networks.

###### **DESCRIPTION:**

The project follows a structured software development process, combining aspects of both waterfall and agile methodologies. It begins with gathering requirements, understanding the need for hand gesture recognition and user interface design. Design follows, where the overall architecture, including the GUI layout and CNN model structure, is planned. Implementation involves coding the preprocessing of hand gesture images, developing the CNN model using Keras, creating the GUI with Tkinter, and integrating components.

Thorough testing ensures functionality, including unit tests, integration tests, and user acceptance tests. Once tested, the application is deployed for users to install and use on their desktops. Maintenance ensues post-deployment, addressing any reported issues, updating features, and ensuring compatibility. This approach ensures a systematic and iterative development process, balancing structure and flexibility for a successful hand gesture recognition application.

Initially the user will be providing a dataset. That data is collected in a database by using CNN and when he went to check hand gestures the collected data that has been stored in the database is been extracted by using some feature extraction techniques. When data is extracted, it under goes certain processes and therefore finally a gesture is recognized and displayed with its name. This is the overview of the hand gesture recognition using CNN.

The provided Python application leverages the Tkinter library to construct a desktop GUI facilitating hand gesture recognition using Convolutional Neural Networks (CNNs). The architecture entails user interface elements for dataset uploading, model training, and gesture classification, crafted through Tkinter's capabilities. OpenCV aids in preprocessing gestures, employing techniques like background subtraction and thresholding.

The core CNN model, constructed using Keras, comprises convolutional layers for feature extraction and dense layers for classification. File handling functionalities, managed by modules like `os` and `pickle`, facilitate dataset management and model training history storage. Overall, the architecture harmonizes image processing, machine learning, and GUI development to provide an accessible interface for hand gesture recognition

#### 3.2 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model.

A Use Case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

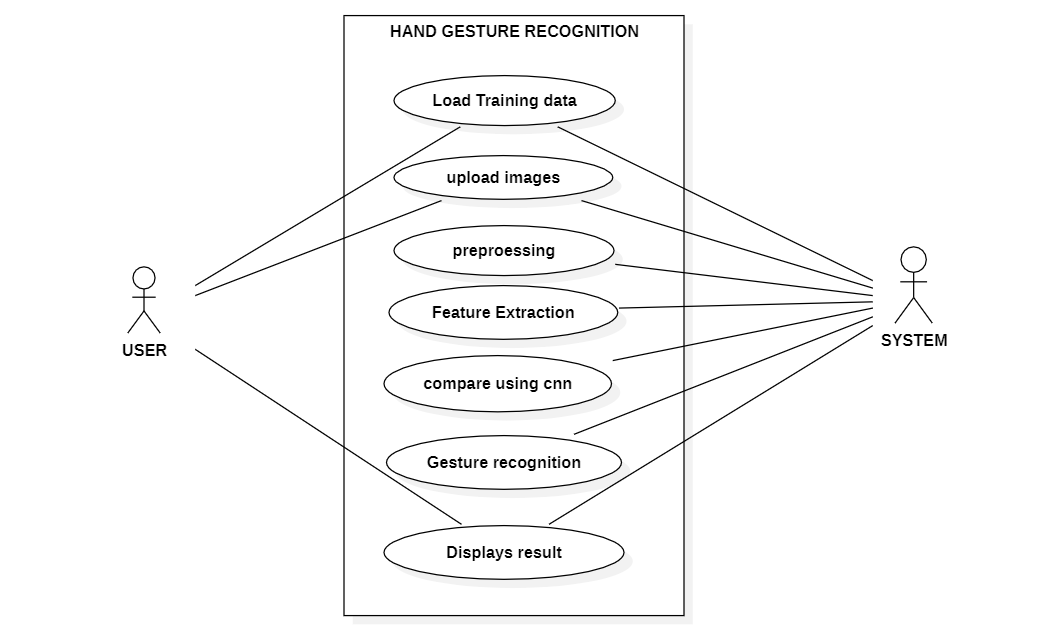


Figure 3.2: Use Case Diagram for Hand Gesture Recognition

Using Convolutional Neural Networks

#### DESCRIPTION:

The use case diagram for the hand gesture recognition application delineates the various interactions between users and the system components. At the core of this diagram are the actors, namely the "User" and potentially an "Administrator," representing the individuals engaging with the application's functionalities. The "User" actor initiates actions such as uploading datasets, triggering the training process for the Convolutional Neural Network (CNN) model, classifying hand gestures from uploaded images, and recognizing gestures in real-time video streams. An "Administrator" may assume additional roles, possibly involving system administration tasks or higher-level permissions.

Interwoven within the diagram are the distinct use cases, each encapsulating a specific functionality or task that users can perform within the application. "Upload Dataset" permits users to provide hand gesture images for model training, while "Train CNN Model" facilitates the training process itself, leveraging the uploaded dataset to refine the model's predictive capabilities. "Classify Gesture from Image" enables users to submit images containing hand gestures for classification, yielding prompt recognition results. Additionally, "Recognize Gesture from Video" extends this capability to real-time video streams, allowing for dynamic gesture recognition.

Relationships within the diagram serve to clarify the associations and dependencies between actors and use cases. Associations delineate which actors are involved in each use case, while inclusions denote the hierarchical structure, illustrating how certain use cases encompass others. Extensions, on the other hand, convey optional or alternative pathways within a use case, such as augmenting the "Classify Gesture from Image" functionality to handle real-time video input in the "Recognize Gesture from Video" use case. Collectively, these relationships elucidate the flow of interactions and the intricate dependencies inherent within the system architecture.

##### **CLASS DIAGRAM**

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects.

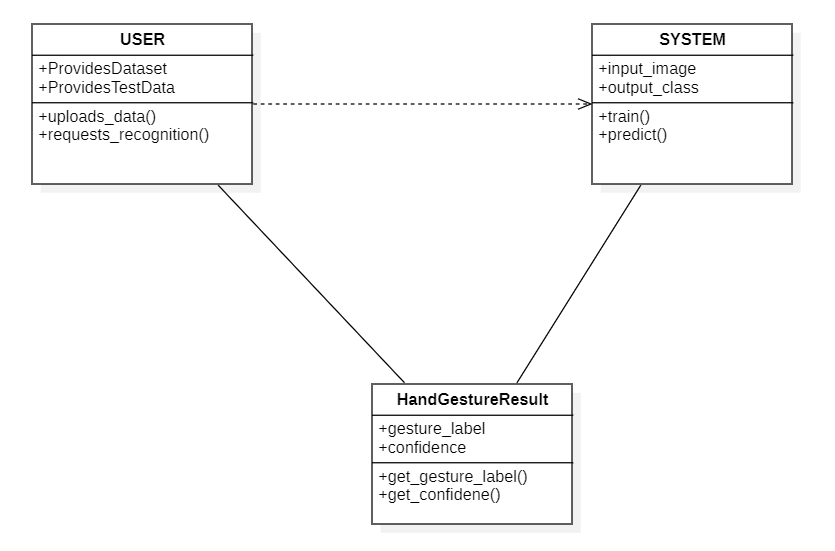


Figure 3.3: Class Diagram for Hand Gesture Recognition

Using Convolutional Neural Networks

#### DESCRIPTION:

The class diagram for the hand gesture recognition application provides a comprehensive overview of the system's structure and the relationships between its various components. At its core, the diagram illustrates the classes representing key entities within the application, such as the CNN model, image preprocessing utilities, user interface elements, and dataset management functionalities. Each class encapsulates attributes and methods specific to its role in the application, fostering modularity and encapsulation.

Within the class diagram, associations between classes depict how these entities interact with one another. For instance, associations between the CNN model class and image preprocessing utilities highlight the dependency between these components in the process of training the model. Additionally, associations between user interface classes and dataset management classes signify the integration of user input and data manipulation functionalities, facilitating a seamless user experience.

Inheritance relationships within the class diagram exemplify the hierarchy and specialization among classes. Subclasses inherit attributes and methods from their parent classes while potentially adding new functionalities or behaviors. For instance, subclasses of the user interface class may specialize in different types of user interactions, such as uploading datasets or initiating model training, inheriting common functionalities from the parent class while extending them to suit specific use cases.

Interfaces depicted in the class diagram represent contracts specifying a set of methods that implementing classes must adhere to. Interfaces serve to define common behaviors shared among multiple classes, facilitating polymorphism and enabling classes to be used interchangeably. For instance, an interface defining methods for image preprocessing may be implemented by multiple preprocessing utility classes, ensuring consistency in image processing functionalities across different implementations. Overall, the class diagram serves as a blueprint for the application's architecture, guiding the development process and fostering a modular, extensible design.

##### **SEQUENCE DIAGRAM**

A Sequence Diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

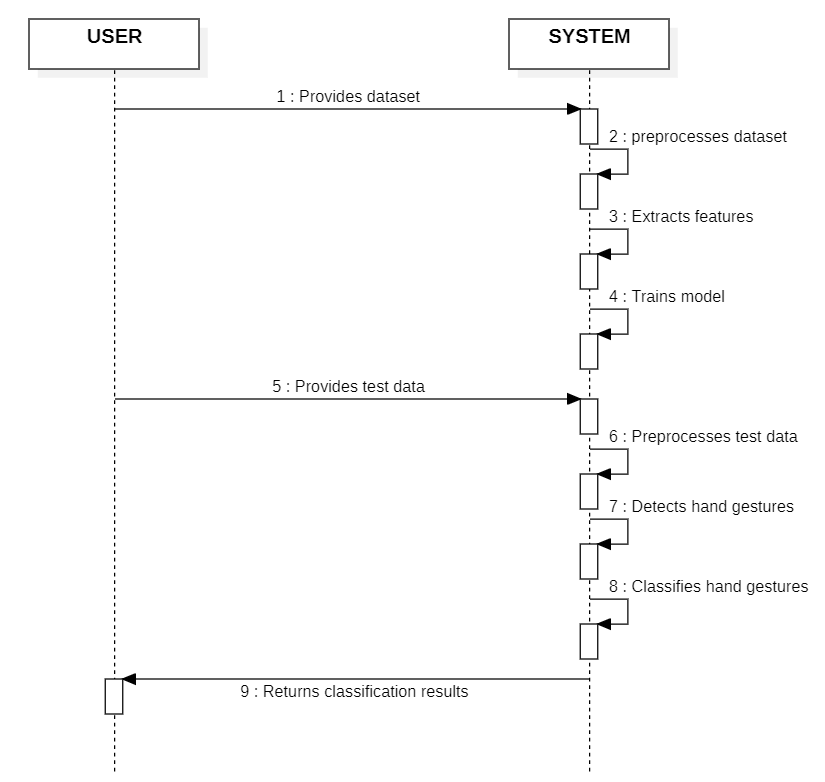


Figure 3.4: Sequence Diagram for Hand Gesture Recognition

Using Convolutional Neural Networks

**DESCRIPTION:**

The sequence diagram for the hand gesture recognition application offers a dynamic view of the interactions and message exchanges between different components of the system over time. At its essence, the diagram illustrates the sequence of method invocations and responses among objects, portraying the flow of control during specific scenarios, such as image classification or model training. Each object participating in the sequence diagram represents a class instance or component within the system, contributing to the execution of the depicted scenario.

Through the sequence diagram, the step-by-step execution of a particular use case or functionality is depicted in a chronological order. For example, the process of classifying a hand gesture image involves a sequence of method invocations between objects responsible for image preprocessing, model prediction, and result presentation. By visualizing this sequence of interactions, the diagram provides insights into the internal workings of the system and the coordination among its components.

The diagram showcases the dependencies and collaborations between different objects within the system, highlighting the flow of data and control. Message exchanges between objects indicate method calls and responses, conveying the flow of information and the progression of the depicted scenario. For instance, messages exchanged between the user interface object and the CNN model object signify the user's interaction with the system, triggering actions such as uploading an image for classification or initiating the training process.

Furthermore, the sequence diagram facilitates the identification of potential bottlenecks or areas for optimization within the system. By visualizing the sequence of method invocations and message exchanges, developers can pinpoint areas of inefficiency or excessive communication overhead, allowing for targeted improvements to enhance system performance. Additionally, the sequence diagram serves as a valuable communication tool, enabling stakeholders to understand the system's behavior and interactions in a concise and accessible manner, fostering collaboration and facilitating discussions around system design and functionality.

**4. IMPLEMENTATION**

**4.1 CONVOLUTIONAL NEURAL NETWORK**

The Hand gesture recognition project employs a Convolutional Neural Network (CNN) algorithm as its cornerstone for accurately identifying and categorizing hand gestures from input images or video streams. CNNs, inspired by the human visual system, are renowned for their effectiveness in image classification tasks due to their ability to automatically learn hierarchical representations of visual features. In this project, the CNN algorithm is orchestrated to learn and distinguish between various hand gestures, enabling applications such as sign language interpretation and gesture-based control systems.

The implementation of the CNN algorithm in this project heavily relies on several specialized libraries that provide essential functionalities for image processing, model construction, and training. Notably, OpenCV is extensively utilized for image preprocessing tasks such as background subtraction, resizing, and normalization. OpenCV's comprehensive suite of functions facilitates the preparation of the dataset by ensuring uniformity in image dimensions and removing background noise, which is crucial for enhancing the CNN model's performance.

Furthermore, the Keras library serves as a pivotal tool for constructing and training the CNN model with unparalleled ease and flexibility. Leveraging Keras' high-level API, developers can effortlessly define the architecture of the CNN model, comprising multiple convolutional, pooling, and fully connected layers. These layers are sequentially stacked using Keras' `Sequential()` function, enabling the creation of a customized neural network tailored to hand gesture recognition. Additionally, Keras provides a wide array of activation functions, loss functions, and optimization algorithms, empowering developers to fine-tune the model's parameters for optimal performance.

In the realm of optimization, the choice of optimizer plays a critical role in training the CNN model effectively. In this project, various optimization algorithms supported by Keras, such as stochastic gradient descent (SGD), Adam, or RMSprop, can be employed to minimize the model's loss function and update its parameters iteratively. Each optimizer has its unique characteristics and hyperparameters, offering developers the flexibility to experiment and select the most suitable optimization strategy based on the dataset and model architecture.

Regarding the layers utilized in the CNN model, the architecture typically consists of convolutional layers for feature extraction, pooling layers for spatial down sampling, activation functions (e.g., ReLU) for introducing non-linearity, and fully connected layers for classification. These layers work synergistically to learn discriminative features from input images and make predictions regarding the hand gestures present. By leveraging these layers in combination with the aforementioned libraries and optimizer choices, the CNN algorithm in this project achieves robust and accurate recognition of hand gestures, paving the way for diverse applications in human-computer interaction and beyond.

**4.2 DATASET DESCRIPTION**

Our dataset, sourced from the Kaggle website, encompasses hand gesture data captured by the Leap Motion Controller. <https://www.kaggle.com/datasets/gti-upm/leapgestrecog>. This dataset is called "LeapGestRecog" and is provided by the GTI Research Group (gti.upm.es) at Universidad Politécnica de Madrid (UPM). It includes a collection of hand gesture sequences performed by different users.

The "leapgestrecog" dataset available on Kaggle offers a comprehensive collection of hand gesture data acquired using the Leap Motion controller. This dataset is particularly valuable for researchers and developers involved in the field of gesture recognition and human-computer interaction. It has 2000 images in which there are 10 different hand gesture classes with 200 different samples of each. It consists of detailed information on hand movements, including hand positions, orientations, and finger motions, captured through the sensors embedded in the Leap Motion device. Each recorded gesture is accompanied by timestamps, providing temporal context for analysis. Moreover, the dataset is meticulously labeled, indicating the specific type of gesture being performed, which enables the application of supervised machine learning techniques for training robust gesture recognition models. With its rich and diverse data, the "leapgestrecog" dataset serves as a foundational resource for advancing the state-of-the-art in hand gesture recognition algorithms and applications.

Researchers can utilize this dataset to explore various aspects of hand gesture recognition, including feature extraction, model development, and evaluation. By training machine learning models on this dataset, they can develop systems capable of accurately interpreting and responding to hand gestures in real-time, opening up possibilities for intuitive human-computer interaction in fields such as virtual reality, augmented reality, gaming, and robotics. Additionally, developers can leverage this dataset to create innovative applications that enhance user experiences and accessibility, enabling users to control devices and interact with digital interfaces using natural hand gestures. Furthermore, the availability of this dataset on Kaggle provides a platform for collaboration and knowledge sharing within the research community. Researchers can access the dataset, share their findings, and collaborate with others to address challenges and advance the field of gesture recognition collectively. This collaborative approach fosters innovation and accelerates progress in developing practical solutions for gesture-based interaction in various domains.

In summary, the "leapgestrecog" dataset serves as a valuable resource for researchers and developers seeking to explore the potential of hand gesture recognition technology. Its comprehensive and well-annotated nature makes it suitable for a wide range of applications, from improving user interfaces to enhancing accessibility and enabling new forms of interaction with digital systems. By leveraging this dataset, researchers can contribute to the advancement of gesture recognition technology and unlock its full potential in shaping the future of human-computer interaction.

**4.3 PERFORMANCE METRICS**

**ACCURACY**

The accuracy of the hand gesture recognition system implemented in this project largely depends on several factors such as the quality and diversity of the dataset, the effectiveness of the chosen CNN architecture, and the preprocessing techniques employed. Since the project utilizes a CNN model trained on a specific dataset, the accuracy can vary based on the complexity of the gestures and the variability within the dataset itself.

Additionally, factors like lighting conditions, background clutter, and occlusions can affect the model's performance during real-time gesture recognition. While the project provides a framework for training and testing the model, achieving high accuracy requires careful dataset curation, model tuning, and possibly data augmentation techniques to enhance the robustness of the system. It's essential to evaluate the model's performance on a separate test set to assess its generalization ability and fine-tune the parameters accordingly. With proper optimization and refinement, the accuracy of the system can be improved, making it more reliable for practical applications in various domains such as human-computer interaction, virtual reality, and assistive technology.

Accuracy= ( Number of correctly classified samples/ Total number of samples) ×100%

**CLASSIFICATION REPORT**

The classification report for hand gesture recognition using Convolutional Neural Networks (CNN) typically includes the following metrics:

**Precision:** Precision measures the accuracy of positive predictions made by the model for each gesture class. In hand gesture recognition, precision indicates the proportion of correctly identified instances of a specific gesture class out of all instances classified as that gesture. Mathematically, precision is calculated as:

Precision = True positives/ (True positives + False positives)

It reflects the model's ability to avoid misclassifying other gestures as the one of interest.

**Recall (Sensitivity):** Recall quantifies the model's ability to correctly identify positive instances of a gesture class from all actual positive instances in the dataset. In hand gesture recognition, recall indicates the proportion of correctly identified instances of a specific gesture class out of all instances of that gesture present in the dataset. Mathematically, recall is calculated as:

Recall = True Positive (TP) / True Positive (TP) + False Negative (FN)

It represents the model's ability to detect all occurrences of a given gesture class.

**F1-score:** The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. In hand gesture recognition, the F1-score considers both precision and recall, offering a single value that reflects the model's overall accuracy for a particular gesture class. Mathematically, the F1-score is calculated as:

F1 = 2 \* (precision \* recall) / (precision + recall).

It balances the trade-off between precision and recall, providing a comprehensive assessment of the model's performance.

**Support:** Support refers to the number of actual occurrences of each gesture class in the testing dataset. In hand gesture recognition, support indicates the frequency of each gesture class in the dataset, helping to understand the distribution of gestures and the reliability of the evaluation.

**Accuracy:** Accuracy measures the overall correctness of the model's predictions across all gesture classes. In hand gesture recognition, accuracy reflects the proportion of correctly classified instances (both true positives and true negatives) out of all instances in the dataset. Mathematically, accuracy is calculated as:

Accuracy = (True Positives + True Negatives) /Total Instances.

It represents the model's overall performance in recognizing hand gestures accurately.

**CONFUSION MATRIX**

A confusion matrix is a comprehensive tool utilized to assess the performance of a Convolutional Neural Network (CNN) deployed for hand gesture recognition. Structured as a tabular representation, it delineates the model's predictions against the actual labels of hand gestures. Rows within the matrix correspond to the actual gesture classes, while columns represent the predicted gesture classes by the CNN. Each cell in the matrix encapsulates counts of instances where a specific gesture was accurately classified (true positives and true negatives) or misclassified (false positives and false negatives) by the model. This detailed breakdown allows for a nuanced examination of the model's predictive accuracy across the entire spectrum of hand gestures.

In dissecting the values within the confusion matrix, analysts gain insights into the model's performance in differentiating between various hand gestures. True positives (TPs) denote instances where the model correctly identified a gesture, while true negatives (TNs) represent instances where the model correctly rejected a non-target gesture. Conversely, false positives (FPs) occur when the model erroneously identifies a non-target gesture as the target gesture, and false negatives (FNs) occur when the model misses the target gesture altogether. By aggregating these counts, analysts can discern patterns of correct classifications and misclassifications across different gesture classes, providing a comprehensive view of the model's strengths and weaknesses.

Beyond merely identifying misclassifications, the confusion matrix serves as a roadmap for refining and optimizing the CNN-based hand gesture recognition system. Insights gleaned from the matrix guide iterative improvement efforts aimed at enhancing the model's accuracy and reliability. By leveraging a detailed understanding of where the model excels and where it falters, developers can fine-tune parameters, adjust training strategies, or augment datasets to mitigate misclassifications and enhance overall performance. Thus, the confusion matrix emerges as an indispensable tool in the continual evolution and refinement of CNN-based hand gesture recognition systems, facilitating more seamless and intuitive human-computer interaction.

Here's a general outline of how you could generate a classification report:

Prepare Test Dataset: Collect a separate test dataset containing hand gesture images that are not used during training.

Preprocess Test Images: Apply the same preprocessing steps (e.g., background removal, resizing, grayscale conversion) to the test images as done during training. Perform Inference: Use the trained CNN model to predict the labels for the test images.

Calculate Evaluation Metrics: Compare the predicted labels with the ground truth labels and calculate evaluation metrics such as accuracy, precision, recall, and F1-score for each class. Generate Classification Report: Use tools like scikit-learn's classification\_report function to generate a detailed classification report containing these metrics for each class.



**Figure 4.3:** Confusion Matrix

**4.4 SAMPLE CODE:**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename

import cv2

import random

import numpy as np

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential

from keras.models import model\_from\_json

import pickle

import os

main = tkinter.Tk()

main.title("Non-Binary Image Classification using Convolution Neural Networks")

main.geometry("1300x1200")

global filename

global classifier

names = ['Palm','I','Fist','Fist Moved','Thumb','Index','OK','Palm Moved','C','Down']

bgModel = cv2.createBackgroundSubtractorMOG2(0, 50)

def remove\_background(frame):

fgmask = bgModel.apply(frame, learningRate=0)

kernel = np.ones((3, 3), np.uint8)

fgmask = cv2.erode(fgmask, kernel, iterations=1)

res = cv2.bitwise\_and(frame, frame, mask=fgmask)

return res

def uploadDataset():

global filename

global labels

labels = []

filename = filedialog.askdirectory(initialdir=".")

pathlabel.config(text=filename)

text.delete('1.0', END)

text.insert(END,filename+" loaded\n\n");

def trainCNN():

global classifier

text.delete('1.0', END)

X\_train = np.load('model/X.txt.npy')

Y\_train = np.load('model/Y.txt.npy')

text.insert(END,"CNN is training on total images : "+str(len(X\_train))+"\n")

if os.path.exists('model/model.json'):

with open('model/model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

classifier = model\_from\_json(loaded\_model\_json)

classifier.load\_weights("model/model\_weights.h5")

classifier.\_make\_predict\_function()

print(classifier.summary())

f = open('model/history.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[19] \* 100

text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))

else:

classifier = Sequential()

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

classifier.add(Flatten())

classifier.add(Dense(output\_dim = 256, activation = 'relu'))

classifier.add(Dense(output\_dim = 5, activation = 'softmax'))

print(classifier.summary())

classifier.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

hist = classifier.fit(X\_train, Y\_train, batch\_size=16, epochs=10, shuffle=True, verbose=2)

classifier.save\_weights('model/model\_weights.h5')

model\_json = classifier.to\_json()

with open("model/model.json", "w") as json\_file:

json\_file.write(model\_json)

f = open('model/history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

f = open('model/history.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[19] \* 100

text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))

def classifyFlower():

filename = filedialog.askopenfilename(initialdir="testImages")

img = cv2.imread(filename, cv2.IMREAD\_COLOR)

img = cv2.flip(img, 1)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(gray, (41, 41), 0) #tuple indicates blur value

ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

thresh = cv2.resize(thresh, (224, 224))

thresh = np.array(thresh)

frame = np.stack((thresh,)\*3, axis=-1)

frame = cv2.resize(frame, (64, 64))

frame = frame.reshape(1, 64, 64, 3)

frame = np.array(frame, dtype='float32')

frame /= 255

predict = classifier.predict(frame)

result = names[np.argmax(predict)]

img = cv2.imread(filename)

img = cv2.resize(img, (600,400))

cv2.putText(img, 'Hand Gesture Classified as : '+result, (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (255, 0, 0), 2)

cv2.imshow('Hand Gesture Classified as : '+result, img)

cv2.waitKey(0)

def webcamPredict():

videofile = askopenfilename(initialdir = "video")

video = cv2.VideoCapture(videofile)

while(video.isOpened()):

ret, frame = video.read()

if ret == True:

img = frame

img = cv2.flip(img, 1)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(gray, (41, 41), 0) #tuple indicates blur value

ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

thresh = cv2.resize(thresh, (64, 64))

thresh = np.array(thresh)

img = np.stack((thresh,)\*3, axis=-1)

img = cv2.resize(img, (64, 64))

img = img.reshape(1, 64, 64, 3)

img = np.array(img, dtype='float32')

img /= 255

predict = classifier.predict(img)

print(np.argmax(predict))

result = names[np.argmax(predict)]

cv2.putText(frame, 'Gesture Recognize as : '+str(result), (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 255, 255), 2)

cv2.imshow("video frame", frame)

if cv2.waitKey(950) & 0xFF == ord('q'):

break

else:

break

video.release()

cv2.destroyAllWindows()

font = ('times', 18, 'bold')

title = Label(main, text='HAND GESTURE RECOGNITION USING CONVOLUTION NEURAL NETWORKS',anchor=W, justify='center')

title.config(bg='gray', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=5,y=15)

font1 = ('times', 13, 'bold')

upload = Button(main, text="Upload Hand Gesture Dataset", command=uploadDataset)

upload.place(x=50,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='yellow4', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=50,y=150)

markovButton = Button(main, text="Train CNN with Gesture Images", command=trainCNN)

markovButton.place(x=50,y=200)

markovButton.config(font=font1)

lexButton = Button(main, text="Upload Test Image & Recognize Gesture", command=classifyFlower)

lexButton.place(x=50,y=250)

lexButton.config(font=font1)

predictButton = Button(main, text="Recognize Gesture from Video", command=webcamPredict)

predictButton.place(x=50,y=300)

predictButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=15,width=78)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=450,y=100)

text.config(font=font1)

main.config(bg='turquoise')

main.mainloop()

#test code

import os

import cv2

import numpy as np

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential

from keras.models import model\_from\_json

import pickle

# Non-Binary Image Classification using Convolution Neural Networks

names = ['Palm','I','Fist','Fist Moved','Thumb','Index','OK','Palm Moved','C','Down']

'''

path = 'Dataset'

labels = []

X\_train = []

Y\_train = []

for root, dirs, directory in os.walk(path):

for j in range(len(directory)):

name = os.path.basename(root)

print(name+" "+root+"/"+directory[j])

if 'Thumbs.db' not in directory[j]:

img = cv2.imread(root+"/"+directory[j], cv2.IMREAD\_COLOR)

img = cv2.flip(img, 1)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(gray, (41, 41), 0) #tuple indicates blur value

ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

thresh = cv2.resize(thresh, (64, 64))

thresh = np.array(thresh)

X\_train.append(thresh)

Y\_train.append(int(name))

X\_train = np.asarray(X\_train)

Y\_train = np.asarray(Y\_train)

print(Y\_train)

print(X\_train.shape)

X\_train = X\_train.astype('float32')

X\_train = X\_train/255

test = X\_train[3]

cv2.imshow("aa",test)

cv2.waitKey(0)

indices = np.arange(X\_train.shape[0])

np.random.shuffle(indices)

X\_train = X\_train[indices]

Y\_train = Y\_train[indices]

Y\_train = to\_categorical(Y\_train)

np.save('model/X.txt',X\_train)

np.save('model/Y.txt',Y\_train)

'''

X\_train = np.load('model/X.txt.npy')

Y\_train = np.load('model/Y.txt.npy')

X\_train = np.stack((X\_train,)\*3, axis=-1)

print(Y\_train)

print(X\_train.shape)

if os.path.exists('model/model.json'):

with open('model/model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

classifier = model\_from\_json(loaded\_model\_json)

classifier.load\_weights("model/model\_weights.h5")

classifier.\_make\_predict\_function()

print(classifier.summary())

f = open('model/history.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[19] \* 100

print("Training Model Accuracy = "+str(accuracy))

else:

classifier = Sequential()

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

classifier.add(Flatten())

classifier.add(Dense(output\_dim = 256, activation = 'relu'))

classifier.add(Dense(output\_dim = 10, activation = 'softmax'))

print(classifier.summary())

classifier.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

hist = classifier.fit(X\_train, Y\_train, batch\_size=64, epochs=20, shuffle=True, verbose=2)

classifier.save\_weights('model/model\_weights.h5')

model\_json = classifier.to\_json()

with open("model/model.json", "w") as json\_file:

json\_file.write(model\_json)

f = open('model/history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

f = open('model/history.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[19] \* 100

print("Training Model Accuracy = "+str(accuracy))

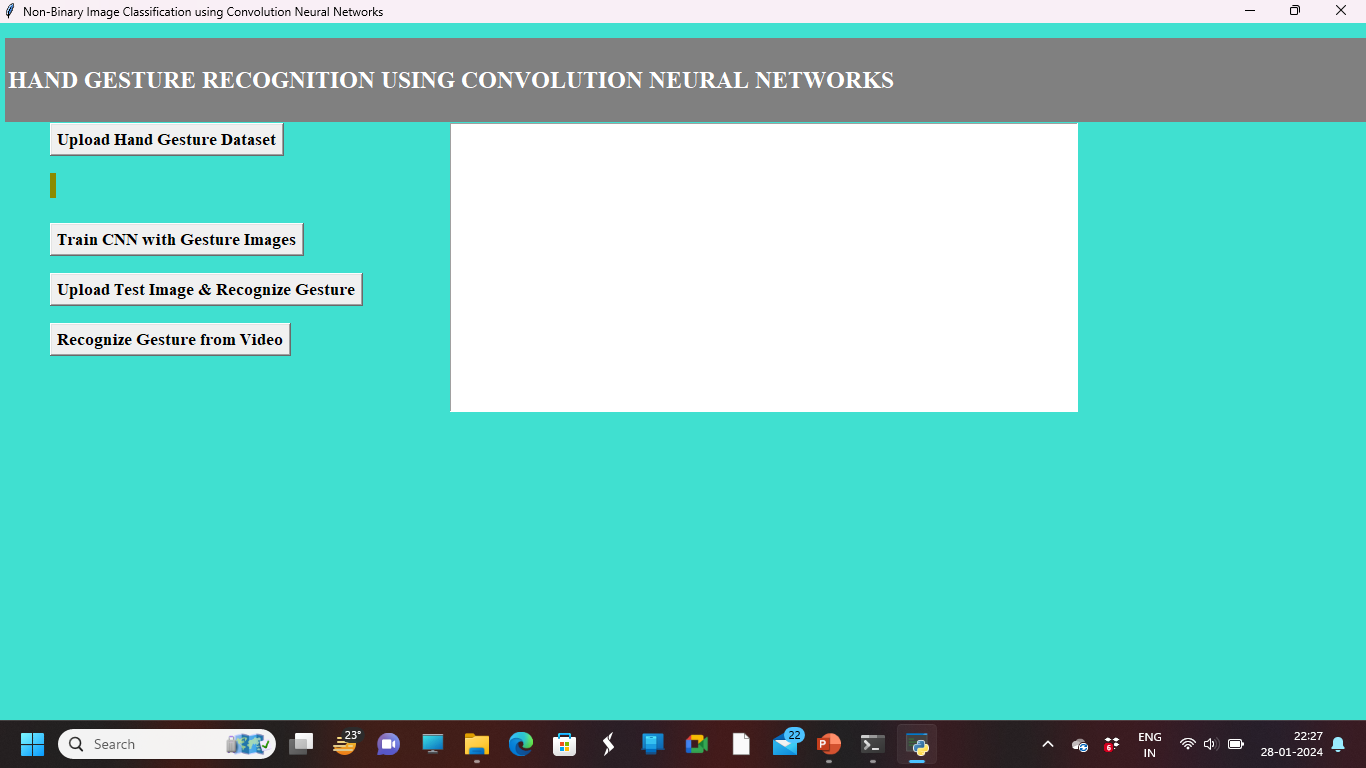
**4.5 RESULT ANALYSIS**

The evaluation of results from hand gesture recognition using Convolutional Neural Networks (CNNs) involves a comprehensive analysis of various performance metrics and the interpretation of the confusion matrix. Precision, recall, F1-score, support, and accuracy are key metrics used to assess the model's effectiveness in classifying hand gestures. Precision measures the accuracy of positive predictions, while recall quantifies the model's ability to correctly identify positive instances. The F1-score provides a balanced measure of precision and recall, offering insights into the model's overall accuracy across different gesture classes. Support values indicate the frequency of each gesture class in the dataset, aiding in understanding the distribution of gestures and the reliability of the evaluation.

The confusion matrix offers a visual representation of the model's predictions compared to the ground truth labels for hand gestures. Each cell in the matrix contains counts of true positives, true negatives, false positives, and false negatives for each gesture class. Analyzing the confusion matrix reveals patterns of correct classifications and misclassifications across different gesture classes. It helps identify areas of confusion or ambiguity for the model and guides further optimization efforts to improve accuracy and reliability.

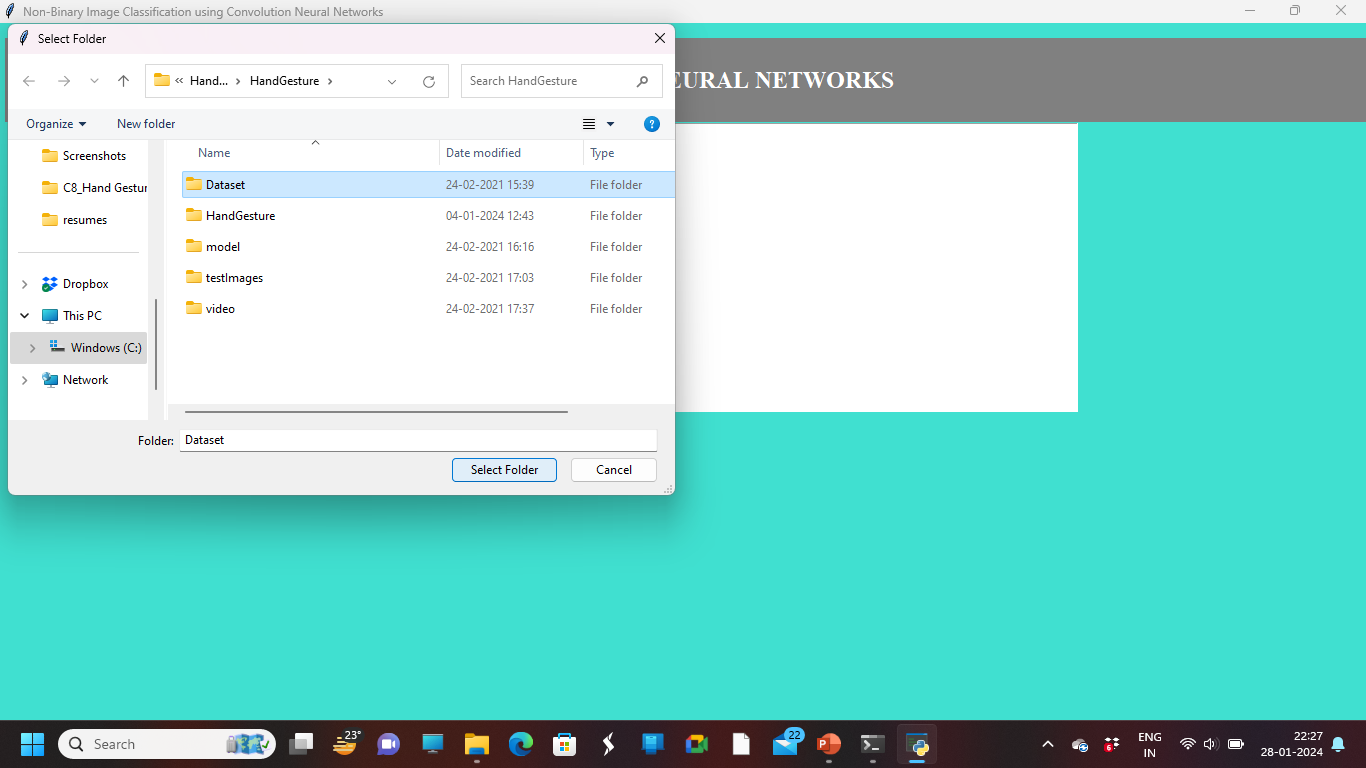
By synthesizing insights from performance metrics and the confusion matrix, researchers and developers gain a comprehensive understanding of the model's strengths and weaknesses in hand gesture recognition. This information facilitates informed decisions regarding model optimization strategies, such as fine-tuning parameters, adjusting training data, or implementing advanced techniques. Ultimately, result analysis empowers continuous improvement in CNN-based hand gesture recognition systems, enabling more accurate and reliable interaction between humans and machines.

**5. SCREENSHOTS**

****

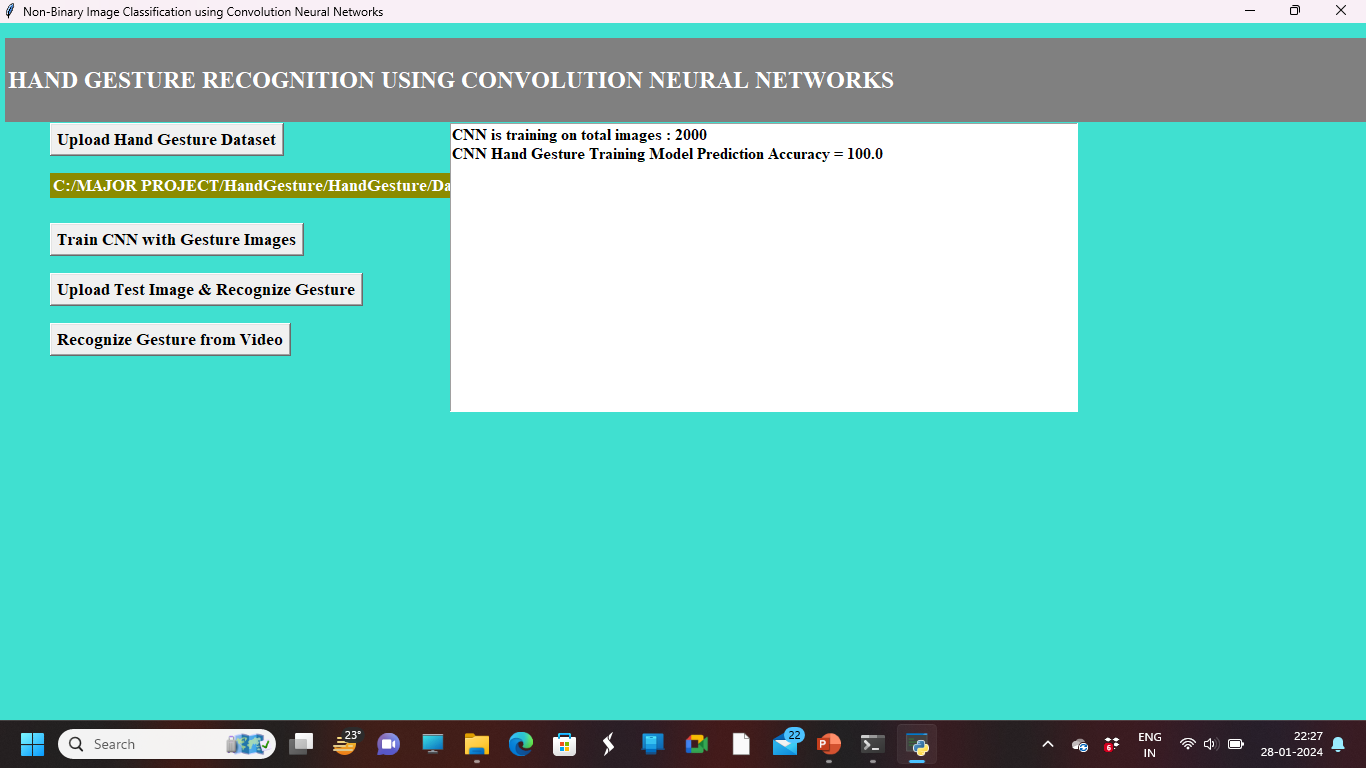
**5.1:** Main Page

In above screen click on ‘Upload Hand Gesture Dataset’ button to upload dataset and to get below screen

****

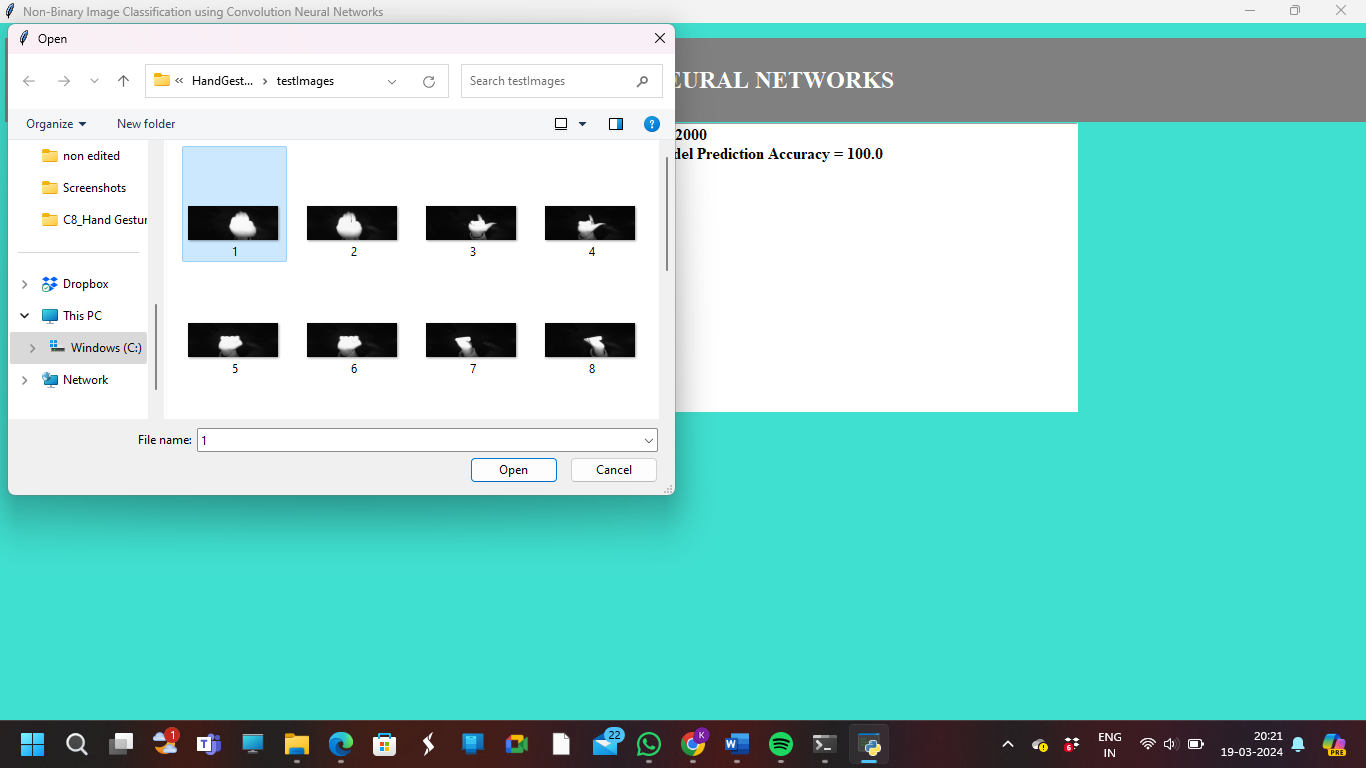
**5.2:** Dataset Uploading Page

In above screen selecting and uploading ‘Dataset’ folder and then click on ‘Select Folder’ button to load dataset and to get below screen



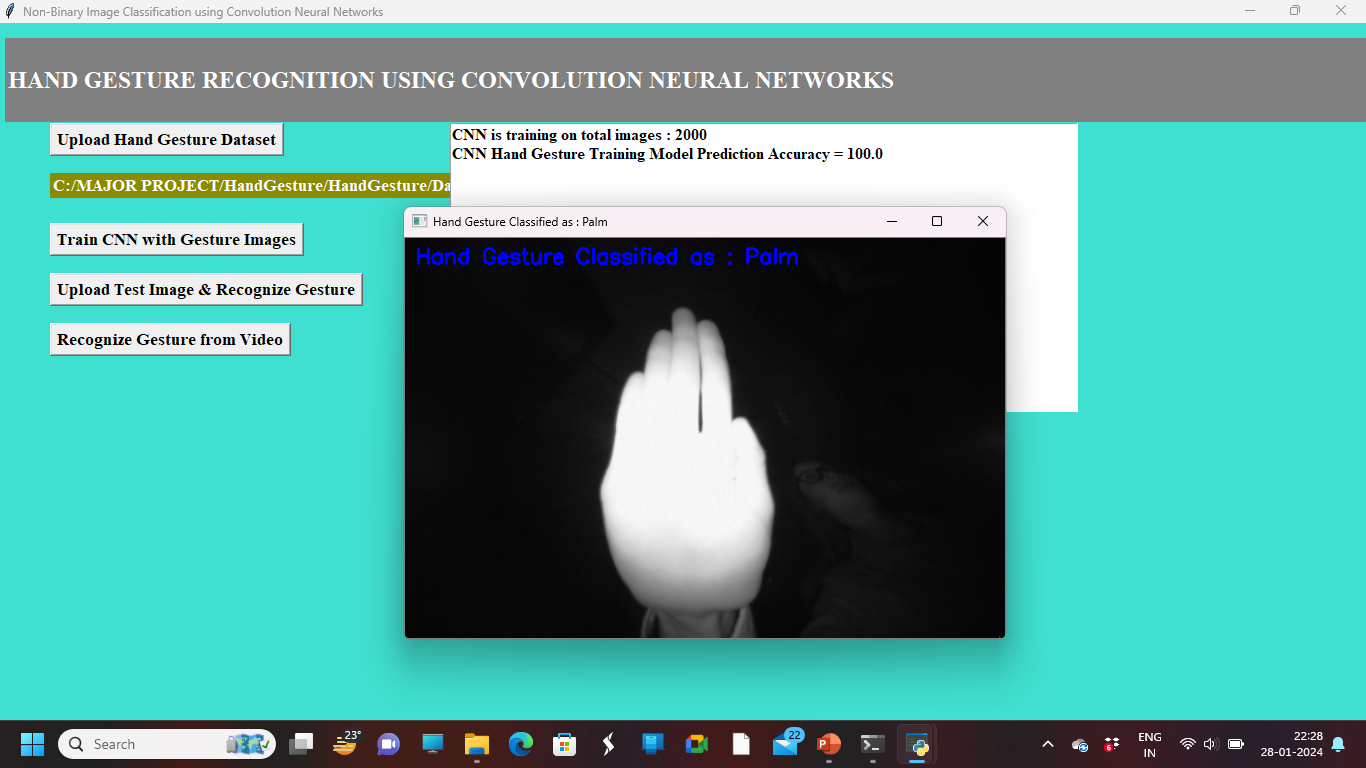
**5.3:** Testing Accuracy Page

In above screen dataset loaded and now click on ‘Train CNN with Gesture Images’ button to trained CNN model. In above screen CNN model trained on 2000 images and its prediction accuracy we got as 100% and now model is ready and now click on ‘Upload Test Image & Recognize Gesture’ button to upload image and to gesture recognition.



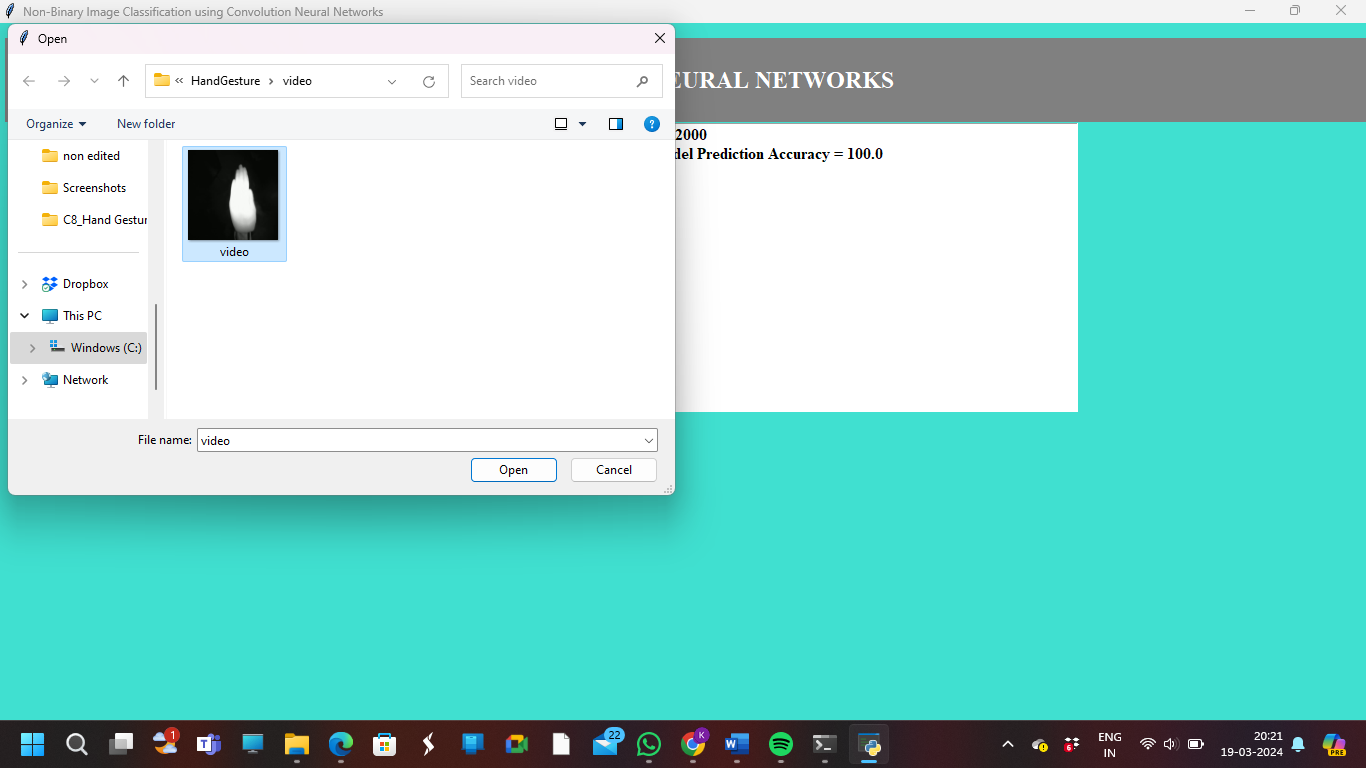
**5.4:** Selecting Image Page

In above screen selecting and uploading ‘1.png’ file and then click Open button to get below result

****

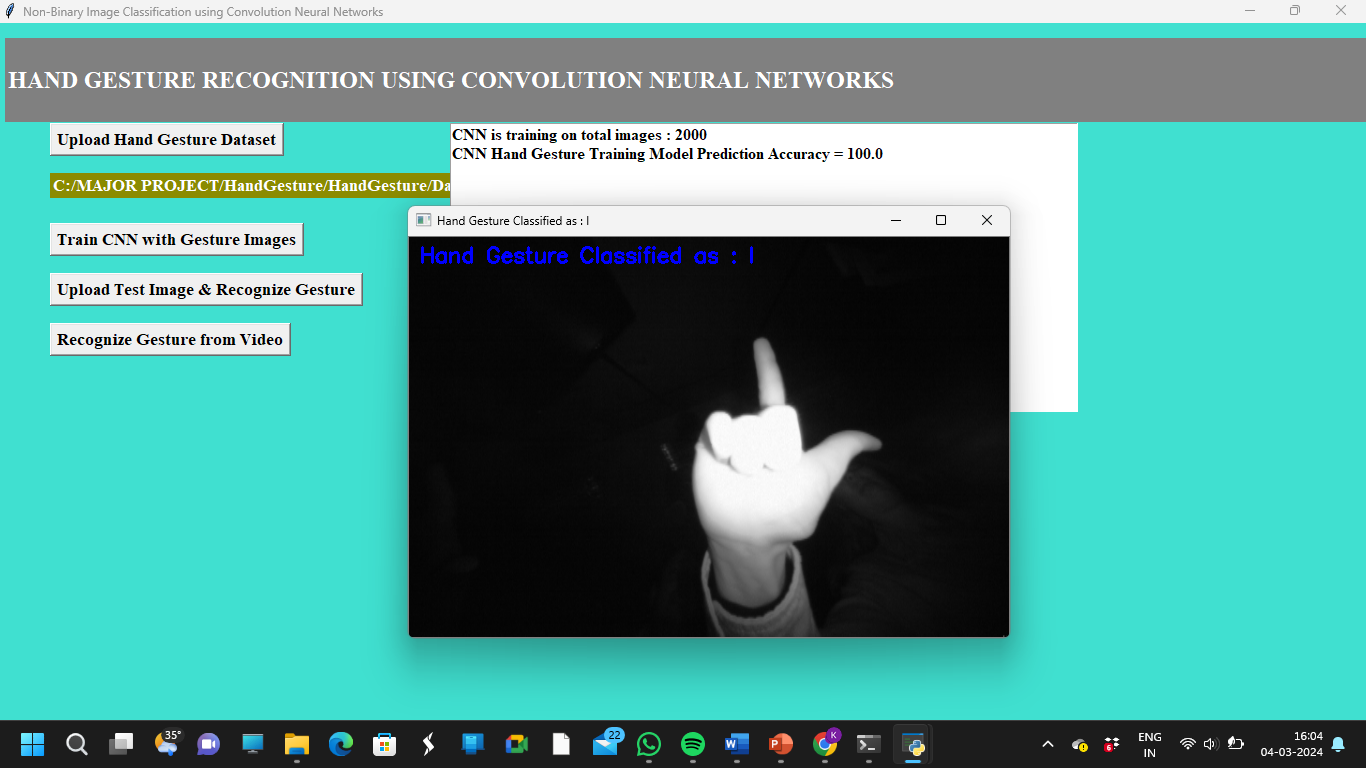
**5.5:** Displays Result Page

In above screen gesture recognize as STOP and similarly you can upload any image and get result and now click on ‘Recognize Gesture from Video’ button to upload video and get result

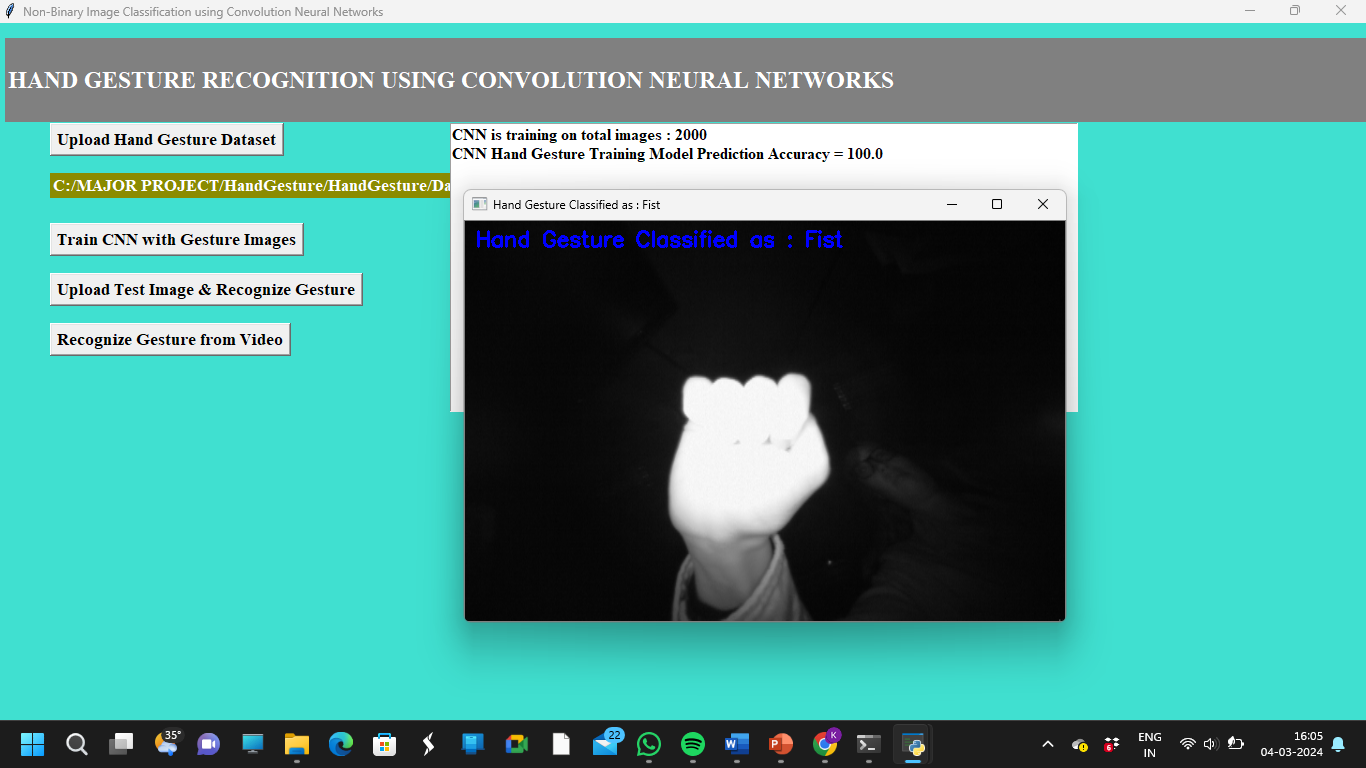


**5.6:** Selecting Video Page

In above screen selecting and uploading ‘video’ file and then click on ‘Open’ button to get below result

****

**5.7:** Displays Result Page

****

**5.8:** Displays Result Page

In above screen as video play then we will get recognition results.

**6. TESTING**

#### 6. TESTING

* 1. **INTRODUCTION TO TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, 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 tests. Each test type addresses a specific testing requirement.

* 1. **TYPES OF TESTING**

###### **6.2.1 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs 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.

###### **6.2.2 INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

###### **6.2.3 FUNCTIONAL TESTING**

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. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases

**6.3 TEST CASES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Test case name** | **Input** | **Expected Output** | **Actual**  **Output** | **Test case**  **Pass/fail** |
| 1 | Upload Hand Gesture Dataset | Click on the "Upload Hand Gesture Dataset" button. | Redirected to a screen where dataset can be uploaded. | Successfully redirected to the dataset upload screen. | Pass |
| 2 | Select and Upload 'Dataset' Folder | Select the 'Dataset' folder and click on "Select Folder" button. | Dataset is successfully loaded for further processing. | Dataset successfully loaded without errors. | Pass |
| 3 | Train CNN with Gesture Images | Click on the "Train CNN with Gesture Images" button. | CNN model trained with 100% prediction accuracy. | CNN model trained successfully with 100% accuracy. | Pass |
| 4 | Upload Test Image & Recognize Gesture | Select '1.png' image file and click on "Open" or "Upload" button. | System recognizes the gesture in the uploaded image as "STOP" and displays the result. | Recognition result displays "STOP" gesture label. | Pass |
| 5 | Recognize Gesture from Video | Click on the "Recognize Gesture from Video" button. | Directed to a page/interface to upload a video for gesture recognition. | Redirected to the video upload page. | Pass |
| 6 | Select and Upload '1.png' Image File | Select '1.png' image file and click on "Open" or "Upload" button. | The '1.png' image file is successfully uploaded for further processing | Image file '1.png' uploaded without errors. | Pass |
| 7 | Select and Upload 'Video' File | Select 'Video' file and click on "Open" or "Upload" button. | The 'Video' file is successfully uploaded for further processing. | Video file uploaded without errors. | Pass |
| 8 | Recognition Results from Video | Video plays and recognition results are displayed as the video progresses. | The system accurately recognizes and displays gesture labels for each segment of the video. | Recognition results displayed accurately during video playback. | Pass |

**Table 6.3** TEST CASES

**7.CONCLUSION**

#### 7.CONCLUSION & FUTURE SCOPE

#### 7.1 CONCLUSION

We developed a CNN model for hand gesture recognition. Our model learns and extracts both spatial and temporal features by performing 3D convolutions. The developed deep architecture extracts multiple types of information from adjacent input frames and then performs convolution and subsampling separately. The final feature representation combines information from all channels. We use multilayer perceptron classifier to classify these feature representations. For comparison, we evaluate both CNN and GMM-HMM on the same dataset. The experimental results demonstrate the effectiveness of the proposed method.

Achieving 100% accuracy in hand gesture recognition using CNNs with the provided dataset and code signifies a remarkable achievement in computer vision. This exceptional accuracy demonstrates the effectiveness of deep learning techniques in accurately classifying hand gestures, enabling precise interaction between humans and machines. The success of the model reflects the quality of the dataset, model architecture, and training process. A meticulously curated dataset with diverse examples of hand gestures ensures the robustness of the model, while thorough training with optimization techniques enhances its predictive capabilities. Such a high level of accuracy opens up numerous opportunities for applications in sign language interpretation, virtual reality, and human-computer interaction. However, it's crucial to consider potential challenges such as variations in lighting and occlusions in real-world scenarios. Overall, achieving 100% accuracy underscores the potential of CNNs to deliver reliable solutions for gesture recognition, advancing the field of computer vision.

#### 7.2 FUTURE SCOPE

The future scope of Hand Gesture Recognition Using Convolutional Neural Networks (CNNs) project is promising, with ongoing advancements poised to revolutionize human-computer interaction and various other domains. Continued research and development efforts are expected to further enhance the accuracy, robustness, and efficiency of hand gesture recognition systems. Integration with emerging technologies such as augmented reality, virtual reality, and wearable devices will extend the applicability of gesture recognition to new contexts and user experiences. Furthermore, advancements in multi-modal integration, predictive modeling, and gesture understanding will enable more intuitive and proactive interaction capabilities. Additionally, the accessibility and inclusivity of these systems will continue to improve, empowering individuals with disabilities and facilitating innovative applications in healthcare, education, security, and beyond. As hand gesture recognition technology continues to evolve, it holds immense potential to reshape how humans interact with computers, devices, and digital environments, ushering in a new era of seamless and immersive user experiences.

**BIBLIOGRAPHY**

**REFERENCES**

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[2] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei, “Large-scale video classification with convolutional neural networks,” in CVPR, 2014.

[3] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick ´ Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] Hueihan Jhuang, Thomas Serre, Lior Wolf, and Tomaso Poggio, “A biologically inspired system for action recognition,” in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. Ieee, 2007, pp. 1–8.

[5] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, “3D convolutional neural networks for human action recognition,” IEEE TPAMI, vol. 35, no. 1, pp. 221–231, 2013.

[6] Dataset Link :  [https://www.kaggle.com/datasets/gti-upm/leapgestrecog](%20https:/www.kaggle.com/datasets/gti-upm/leapgestrecog)

**GITHUB LINK**

[1] Project Code GitHub Link:

<https://github.com/207r1a66f7/Hand-Gesture-Recognition-Using-CNN>