CNN-Based Hand Gesture Recognition System

Monojit Bhattacharya M , Deepan P , Jeral Meijo H , Dharanivel L

Guide: Dr. M. Amutha M.Tech Ph.D ,

Computer Science and Engineering

Bachelor of Engineering
Hindusthan College Of Engineering and Technology
Coimbatore – 641 032

ABSTRACT

Sign language is a system of communication using Visual gestures and signs.Hearing impaired people and the deaf and dumb community use sign language as their only means of communication. Understanding sign language is so much difficult for a normal person. Therefore, the minority group has always faced many difficulties in communicating with the General population. In this research paper, we proposed a new deep learning-based approach to detect sign language, which canremove the barrier of communication between normal and deaf People. To detect real-time sign language first we prepared a Dataset that contains 11 sign words and 26 Alphabets. We used these sign words to Train our customized CNN model. We did some preprocessing in The dataset before the training of the CNN model.. The field of sign language detection has witnessed significant advancements driven by the integration of computer vision, machine learning, and natural languageprocessing techniques. This research focuses on the development of a robust and adaptive system for interpreting sign language gestures, facilitating effective communication between individuals who use sign language and those who may not be familiar with it. Leveraging computer vision, the system captures and analyzes video or image data to recognize intricate hand movements, facial expressions, and other relevant features associated with sign language. Machine learning algorithms, particularly convolutional neural networks (CNNs), are employed to train models that ensure accurate and real-time gesture recognition. Wearable devices, such as gloves or wrist-worn sensors, equipped with motion sensors, offer an alternative approach for capturing and interpreting hand and body movements. The integration of natural language processing techniques further enhances the system's ability to interpret the linguistic aspects of sign language.

CHAPTER 1 1.INTRODUCTION

1.1 INTRODUCTION

In the Era of human vision, where technology and human vision auditory coverage of learning style, our project stands at the forefront of innovation. Titled "Accessing visual information by visually impaired persons by using pytesseract and OpenCV," Phase 2 marks a another module from our initial exploration in Phase 1, where we focused solely on sign language. Now, in Phase 2, our ambition expands as we endeavor to sign language detection, leveraging the powerful capabilities of OpenCV and convolutional neural network

Communication is a vital tool in human existence. it's a basic and effectivemanner of sharing thoughts, feelings and opinions. A considerable fraction of the world's population lacks this many of us are tormented by hearing disorder,speaking impairment or each. A partial or complete inability to listen toin one oreach ear is understood as hearing disorder. On the opposite hand, mute could be an incapacity that impairs speaking and makes the affected folks unableto talk.

If deaf-mute happens throughout childhood, their acquisition ability is hindered and leads to language impairment, additionally referred to as hearing condition. These ailments are a part of the foremost common disabilities worldwide applied mathematics report of physically challenged kids throughout the past decade reveals a rise within the range of neonates born with a defect of handicap .

The primary objective of Phase 2 is to develop a system capable of accurately and efficiently tracking sign language. By leveraging the combined strengths of OpenCV and convolutional neural network.

Throughout this phase, we will focuses on the domain of American Sign Language (ASL) recognition, a field of growing significance in bridging communication gaps for the hearing impaired. The core aim is to develop an efficient ASL recognition system employing advanced machine learning and computer vision techniques. The project's foundation rests on the meticulous collection of a diverse ASL signdataset, necessary for training and validating therecognition model.

Moreover, this project extends pivotal component involves the applicationof Convolutional Neural Networks (CNNs) to automatically extract essential sign language features from images. Recurrent Neural Networks (RNNs), particularly Long Short Term Memory (LSTM) networks, will be incorporated to handle sequential data, crucial for comprehending phrases or sentences in ASL.

The ASL recognition project requires advanced technology, a skilled workforce, structured data resources, and financial backing. These elements are vital to create a system that enhances accessibility and communication for the deaf and hard of hearing community.

 DOI: 10.55041/IJSREM34543

CHAPTER 2

2. OBJECTIVE

The objective of Phase 2 of the project titled "Accessing information forphysically impaired person using sign language detect system" is to extend the capabilities established in Phase 1, which focused solely on sign language, to encompass deaf . This phase aims to delve deeper into the complexities of natural language movement and develop algorithms capable of accurately tracking the hand in real-time using the OpenCV andconvolutional neural network . Specifically, the objectives include:

Gesture Recognition: Develop algorithms to recognize and classify a wide range of sign language gestures, including handshapes, movements, and facial expressions. Ensure the system can distinguish between different signs to accurately interpret the intended message.

Real-time Processing: Implement real-time processing capabilities to enable instantaneous interpretation of sign language gestures as they occur. Minimize latency to create a responsive and natural communication experience. Optimize algorithms and software architecture for quick processing to achieve low latency. Employ hardware acceleration, parallel processing, or edge computing to enhance real-time performance.

Multi-Model Input Recognition: Integrate computer vision models for handand facial feature extraction. Leverage pre-trained models for facial expression recognition to enhance the system's ability to interpret emotions expressed through facial cues.

Integration With Communication platform: Integrate the sign language detection system with existing communication platforms, such as video conferencing tools or messaging apps, to facilitate communication across various contexts. Develop application programming interfaces for seamless integration with popular communication platforms. Ensure compatibility with video conferencing tools, messaging apps, and other platforms to extend the system's usability.

User Interface: Design an interface that displays recognized signs and provides feedback on the accuracy of gestures. Include visual aids, such as highlighting recognized signs or suggesting corrections, to assist users in refining their gestures. Create an intuitive and user-friendly interface that allows both sign language users and those unfamiliar with sign language to interact seamlessly. Include feedback mechanisms to assist users in refining their gestures for better recognition.

Accessibility Standards: Comply with accessibility standards to ensure the system is inclusive and usable for individuals with diverse needs, including those with disabilities. By addressing these objectives, a sign language detection system can contribute to breaking down communication barriers, fostering inclusivity, and providing an effective means of communication .

 DOI: 10.55041/IJSREM34543

CHAPTER 3

3. IDEATION PHASE

3.1 LITERATURE SURVEY

1. Real-time sign language detection using human pose estimation

Author: Amit Moryossef, Ioannis Tsochantaridis, Roee Aharoni.

This paper proposes a real-time sign language detection model, as identify the need for such a case in videoconferencing. We extract optical flow features based on human poseestimation Using a recurrent model directly on the input, we see improvements of up to91% accuracy.

2. Machine Learning Based Real Time Sign Language Detection

Author: P Rishi Sanmitra, VV Sai Sowmya, K Lalithanjana.

In this paper, a real time ML based system was built for the Sign Language Detection using images that have been captured with the help of a PC camera. The main purposeof this project is to design a system for the differently abled people to communicate with others with ease.

3. Fingerspelling detection in american sign language.

Author: Bowen Shi, Diane Brentari, Greg Shakhnarovich, Karen Livescu.

In this paper, which words are signed letter by letter, is an important component of American Sign Language.

4. A review of hand gesture and sign language recognition techniques

Author: Ming Jin Cheok, Zaid Omar, Mohamed Hisham Jaward.

Hand gesture recognition serves as a key for overcoming many difficulties and providing convenience for human life. The ability of machines to understand human activities and their meaning can be utilized in a vast array of applications.

5. Sign language detection

Author: Deep, A., & Chintan Bhatt.

 DOI: 10.55041/IJSREM34543

There are persons whose speaking or hearing abilities are impaired. Communicationpresents a significant barrier for persons with such disabilities

3.2 PROBLEM STATEMENT

In phase 1 of our project, we successfully implemented a visual information system using Ocr and pytesseract, providing accurate and real-timeestimation of visual information. Building upon this foundation, the aim of phase2 is to extend our system to track the sign language in real-time.

The problem definition for this project revolves around addressing the significant communication challenges faced by the deaf and hard of hearing community. These individuals often encounter barriers when trying to communicate with the broader population, as many people are not well versed in sign language. The existing systems and methods for sign language interpretation predominantly rely on human interpreters, which can be expensive, not always readily available, and may compromise privacy. The proposed project seeks to develop a robust and accessible system for real time sign language recognition and interpretation. It aims to create a solution that canunderstand and interpret sign language gestures, signs, and expressions, providing a bridge for communication between the deaf and hard of hearing individuals and those who do not know sign language. The goal is to make communication in sign languageas natural and intuitive as spoken or written language.

CHAPTER 4

4. PROJECT DESIGN PHASE - 2

4.1 PROPOSED SOLUTION

The proposed system is designed as an innovative solution to address the limitations of existing sign language communication methods. It will leverage advanced technologies such as computer vision and machine learning to offer realtimesign language recognition, eliminating the reliance on human interpreters. Users will interact with the system through an intuitive and user friendly interface, which can accommodate sign language gestures, signs, or simple input devices.

One of the key advantages of the proposed system is its comprehensive sign database, which will include a wide range of signs and gestures from varioussign languages. Machine learning algorithms will enable adaptive learning,allowing the system to continuously improve its sign recognition capabilities based on user interactions .Accessibility is a central focus of the proposed system, as it will be available on multiple platforms, including smartphones and tablets. Moreover, users will have the flexibility to customize their sign language settingsto meet their specific needs and ensure privacy. This approach not only enhances accessibility but also promotes inclusivity and empowers users in their communication .The system's cost efficient and sustainable design will reduce theexpenses associated with sign language interpretation while contributing to a more environmentally friendly approach. Furthermore, it will prioritize cultural sensitivity by recognizing the variations in sign languages used across different regions and communities. In summary, the proposed system aims to revolutionizesign language communication, making it more accessible, efficient, and user centric, ultimately breaking down communication barriers for the deaf and hard of hearing community.

The proposed system is to develop an innovative and userfriendly sign language
recognition and interpretation solution that addresses the limitations of the existing systems. This project aims to create a system that offers realtime sign language

recognition and converts it into spoken or written language, bridging the communication gap between the deaf and hard of hearingcommunity and the general population. The primary objectives include enhancingcommunication by providing a seamless and inclusive means of communicationfor individuals who use sign language, prioritizing accuracy and speed in sign language recognition and interpretation, offering flexibility in output options, providing a user friendly interface for people of all ages, ensuring cross platform compatibility for communication in diverse settings, incorporating machine learning capabilities for adaptation to users' signing styles, andstriving to make the technology affordable and scalable for widespread use. This system aims to create a more inclusive and accessible world for the deaf and hard of hearing community, empowering them to engage fully in education, employment, and social interactions.

4.2 SYSTEM ARCHITECTURE

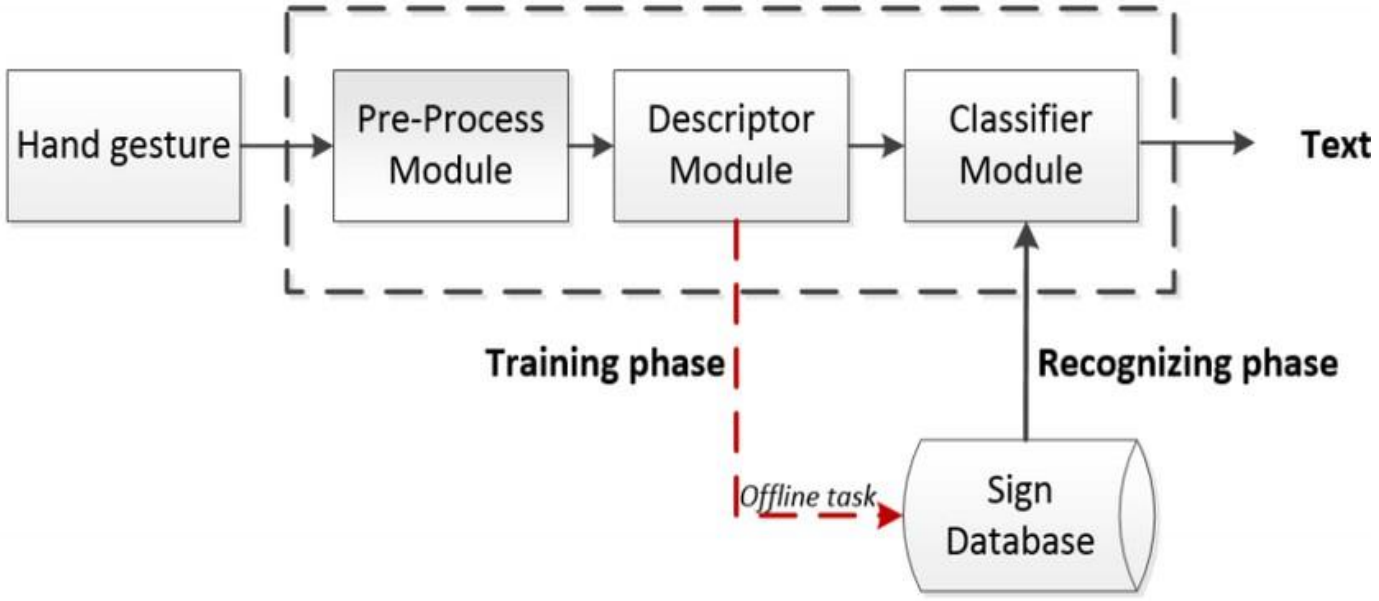


Figure. 4.1 System Architecture

OpenCV:

Video Capture: Utilize OpenCV's video capturing functionalities to access the live camera feed or load pre-recorded video files.
Frame Processing: Extract frames from the video stream and perform preprocessing tasks such as resizing, normalization, and color space adjustmentsto prepare data.

Convolutional Neural Network (CNN):

A Convolutional Neural Network (CNN) is a specialized class of artificialneural networks designed primarily for the analysis of visual data, particularly images and videos. CNNs have gained significant prominence due to their exceptional performance in computer vision tasks. These networks are inspired by the human visual system's ability to perceive and recognize patterns and features in visual information.

American Sign Language:

The ASL recognition system is a crucial stepaimed at optimizing the input data,typically captured through a camera feed. Thisphase involves several key tasks to enhance the quality and suitability of the visual data for subsequent recognitionprocesses.

Preprocessing includes image resizing, which standardizes the dimensions of thecaptured frames. Resizing ensures that all input frames have consistent dimensions, allowing the recognition model to work with uniform image sizes, thereby simplifying the model's design and improving performance. Another important preprocessing step is color conversion.

User Interface:

User Interface (UI) of the ASL recognition system is a critical elementdesigned to facilitate an intuitive and user-friendly experience for its users. Its primary role is to bridge the gap between technology and the users, making ASLcommunication more accessible and inclusive. The UI includes various essentialfeatures, such as real-time video feed, displaying users' signing gestures, and providing instant feedback by showing recognized ASL signs in text form. It alsoincorporates instructional elementsfor users new to ASL, a virtual keyboard for extended conversations, and options for customization to accommodateindividual signing styles.

Performance Optimization:

Optimize algorithms for parallel execution to enhancespeed and efficiency Explore techniques like model quantization or pruning toreduce computational load while maintaining accuracy.

Testing and Validation:

Conduct extensive testing across various scenarios and datasets tovalidate the accuracy, robustness, and real-time performance of the system. This phase is critical to ensure the reliability, functionality, and performance of the system before it is deployedfor real-world use. System testing serves as the final evaluation to confirm that the ASL recognition system meets its intended objectives and operates as expected**.**

CHAPTER 5 5.PROJECT DESIGN PHASE - 2

5.1 SOLUTION REQUIREMENTS

HARDWARE SPECIFICATIONS

- System : Windows 10 or Linux
- Storage : Minimum 512 GB SSD
- Graphics Card : intel ARC A350 or above
- Ram : 8 GB or Above

SOFTWARE SPECIFICATIONS

- Operating system (users) : Windows 10 or Linux
- Coding Language : Python

- Image Format : Jpeg or Webcam format.

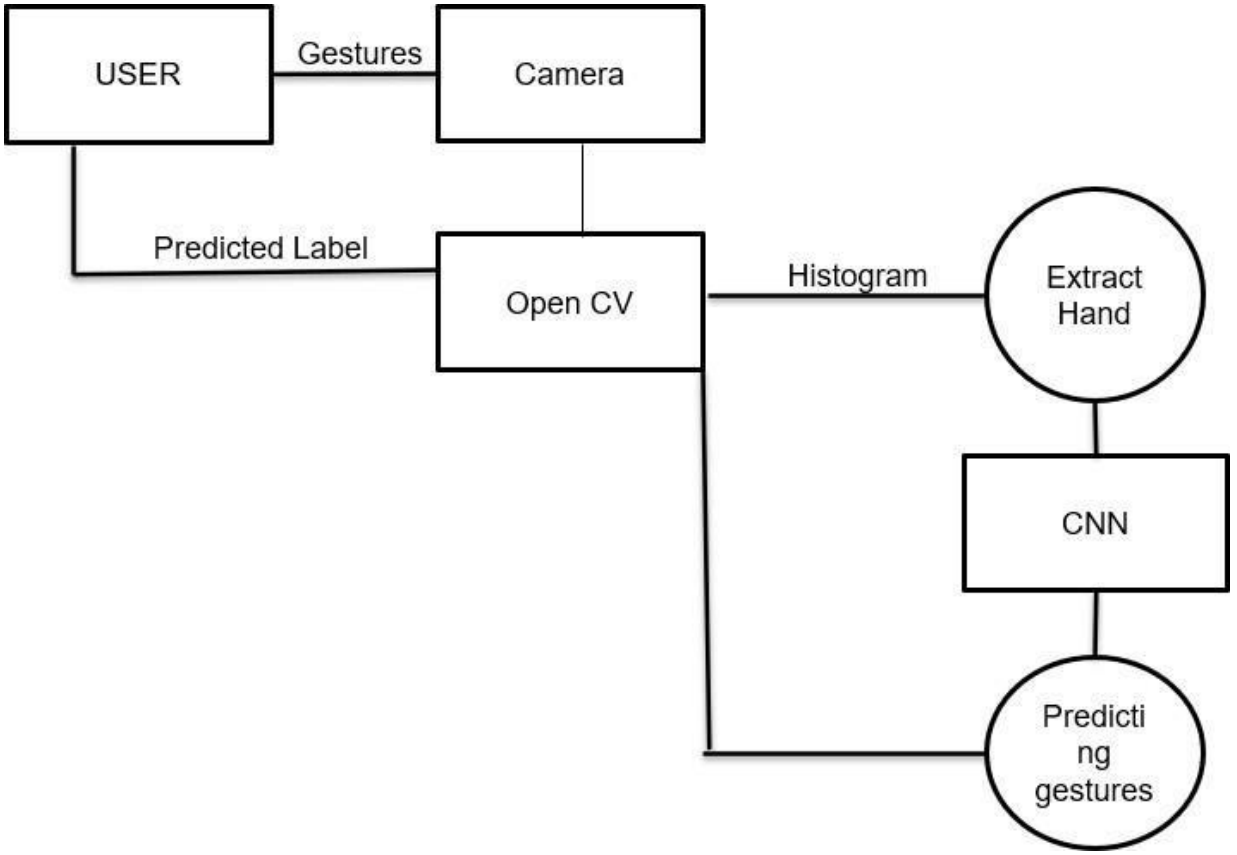
5.2 DATA FLOW DIAGRAMS

5.2.1 Data Flow Diagram Level 0

It is an external system that sends or receives data, communi-cating with the system. Losing access to a system is a source of information and a destination.

Figure. 5.1 Data Flow Diagram Level 0

5.2.2 Data Flow Diagram Level 1

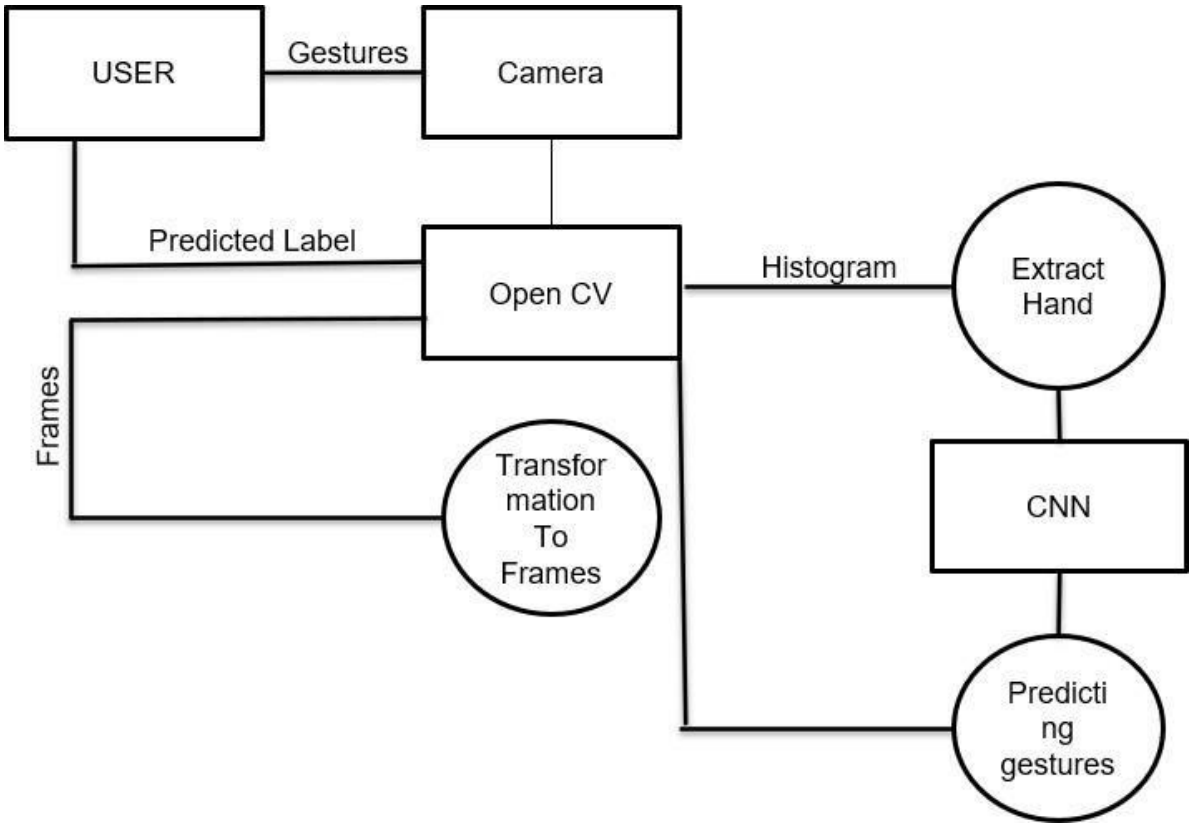


 DOI: 10.55041/IJSREM34543

A dataflow represents a package of information flowing between two objects in the data- flow diagram, Data flows are used to model the flow of information into the system, out of the system and between the elements within the system.

Figure. 5.2 Data Flow Diagram Level 1

5.2.3 Data Flow Diagram Level 2



These are the files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label.

Figure. 5.3 Data Flow Diagram Level 2

 DOI: 10.55041/IJSREM34543

5.3 TECHNOLOGY STACK

5.3.1 PYTHON PROGRAM

Python is a powerful and versatile programming language known for its simplicity and readability. It provides an elegant and concise syntax that makesit easy to write and understand code. Python's design philosophy emphasizes code readability, which allows developers to express ideas more clearly and focus on solving problems rather than getting lost in complex syntax. This readability also makes Python an excellent language for beginners, as it reduces the learning curve and enables them to quickly grasp programming concepts.

One of Python's greatest strengths lies in its vast ecosystem of libraries and frameworks. The Python Package Index (PyPI) hosts a multitude of third- party packages that extend Python's functionality in various domains, includingdata science, web development, machine learning, and more. Popular libraries like NumPy, pandas, and scikit-learn empower data scientists and analysts to efficiently manipulate and analyze data. Django and Flask provide powerful frameworks for web development, while TensorFlow and PyTorch offer robust tools for building and training deep learning models. Python's extensive library support fosters productivity and enables developers to leverage existing solutions, saving time and effort in developing complex applications.

Overall, Python's simplicity, readability, and extensive ecosystem make it a preferred language for a wide range of applications. Whether you're a beginner or an experienced developer, Python's flexibility and community support offer arewarding programming experience that encourages efficient development .

5.3.2 TENSORFLOW

Tensor Flow is an open-source machine learning framework developed by Google. It has gained widespread popularity due to its versatility and powerful capabilities in building and deploying machine learning models. TensorFlow provides a comprehensive ecosystem that supports a wide range of tasks, including neural networks, deep learning, natural language processing, and computer vision. Its flexible architecture allows developers to create complex models with ease, thanks to its extensive collection of pre-built operations and layers. TensorFlow's computational graph abstraction enables efficient execution of operations on both CPUs and GPUs, optimizing performance and scalability.

Another notable feature of TensorFlow is its compatibility with different programming languages. While its primary interface is in Python, TensorFlow provides bindings for other languages such as C++, Java, and JavaScript, allowing developers to work with TensorFlow in their preferred language. Additionally, TensorFlow's high-level APIs, such as Keras, provide a user- friendly interface for quickly prototyping and building machine learning models. With its robust community support, TensorFlow has become a go-to framework for researchers, practitioners, and enthusiasts seeking to develop cutting-edge machine learning applications.

In recent years, TensorFlow has continued to evolve with advancements such as TensorFlow Extended (TFX) for end-to-end machine learning pipelines,TensorFlow Lite for deploying models on mobile and embedded devices, and TensorFlow.js for running models in web browsers. With its wide range of capabilities, extensive documentation, and an active community, TensorFlow remains a leading choice for building and deploying machine learning models

 DOI: 10.55041/IJSREM34543

atscale.

5.3.3 PY TORCH

PyTorch is a popular open-source deep learning framework known for its dynamic computational graph, flexibility, and ease of use. Developed by Facebook's AI Research (FAIR) team, PyTorch has gained significant traction inthe machine learning community. One of its standout features is its dynamic nature, which allows for intuitive and efficient model development. PyTorch enables developers to construct and modify computational graphs on the fly, making it easier to debug, experiment, and iterate on models. This dynamic approach, combined with Pythonic syntax, empowers researchers and developersto express complex ideas and implement custom operations with ease.

PyTorch provides a range of tools and libraries that make deep learning development accessible to both beginners and experts. Its TorchScript feature allows users to export trained models into a production-ready format, enabling seamless deployment across various platforms. The PyTorch ecosystem also includes popular libraries like torchvision for computer vision tasks and torchaudio for audio processing, further expanding its capabilities. With its strong community support and active development, PyTorch continues to evolvewith new features and advancements, making it a preferred choice for researchers, academics, and industry professionals in the field of deep learning.

5.3.4 NumPy

NumPy, short for Numerical Python, is a fundamental library for scientific computing in Python. It provides a powerful array object and a collection of mathematical functions that allow efficient manipulation and computation on large multi-dimensional arrays. NumPy's core data structure, thendarray (N-dimensional array), is optimized for speed and memory efficiency, making it a crucial building block for numerical computations. With NumPy, complex mathematical operations, such as linear algebra computations, Fourier transforms, and random number generation, can be performed efficiently, offering a solid foundation for various scientific and data analysis tasks.

The simplicity and ease of use of NumPy make it a go-to library formany Python programmers and data scientists. NumPy's intuitive and expressivesyntax enables concise and readable code, facilitating faster development and easier collaboration. It also plays a crucial role in enabling the integration of other libraries and frameworks in the scientific Python ecosystem. For example,NumPy arrays serve as the data exchange format between different libraries, such as pandas for data manipulation and scikit-learn for machine learning.

5.3.5 OPEN CV

OpenCV (Open Source Computer Vision Library) is a widely used open- source computer vision and image processing library. It provides a vast collection of algorithms and functions that enable developers to build applications for tasks such as image and video processing, object detection and tracking, facial recognition, and more. OpenCV supports a wide range of programming languages, including Python, C++, and Java, making it accessibleto developers across different platforms.

One of the key strengths of OpenCV is its extensive set of pre-built computer vision algorithms. It offers a rich toolbox of functions for tasks like image filtering, feature detection, geometric transformations, and camera calibration. OpenCV also includes machine learning capabilities, providing access to popular techniques like support vector machines (SVM), decision trees, and neural networks. The library's cross-platform nature and support for various hardware accelerators, such as GPUs, enable efficient execution of computationally intensive computer vision algorithms on diverse devices.

With its active development community, comprehensive documentation, and a wealth of code examples, OpenCV has become the go-to library for computer vision tasks. Its versatility and robustness have made it the backbone of

 DOI: 10.55041/IJSREM34543

numerous real-world applications, spanning industries like robotics, automotive, healthcare, surveillance, and augmented reality. From basic image processing operations to advanced computer vision tasks, OpenCV empowers developers with the tools they need to analyze and manipulate visual data effectively.

5.3.6 CONVOLUTIONAL NEURAL NETWORK

A Convolutional Neural Network (CNN) is a specialized class of artificialneural networks designed primarily for the analysis of visual data, particularly images and videos. CNNs have gained significant prominence due to theirexceptional performance in computer vision tasks. These networks are inspired by the human visual system's ability to perceive and recognize patterns and features in visual information. The fundamental components of a CNN include convolutional layers, pooling layers, activation functions, fully connected layers,loss functions, backpropagation, dropout, and batch normalization.

Convolutional layers play a crucial role in extracting meaningful features from the input data, such as edges, textures, and object parts. Pooling layers reduce the spatial dimensions of the feature maps, enabling the network to be translation- invariant. Non-linear activation functions, like ReLU, introduce non-linearity into the network, making it capable of modeling complex relationships.Fully connected layers are responsible for processing high level features and making predictions. During training, backpropagation adjusts the network's.

IMAGE INPUT LAYER:

The image input layer serves as the entry point for convolutional neural networks (CNNs). It takes in raw or preprocessed image data and sets the dimensions of input images, typically as 224x224 pixels with three color channels(RGB). This layer is crucial for maintaining consistent and standardized input through processes like data normalization, which helps the network handle variations in input intensity. Additionally, it can apply data augmentation techniques to enhance the network's robustness. The image input layer allows processing multiple images in parallel when dealing with batches, forming the foundation for efficient image analysis in CNNs.

CONVOLUTIONAL LAYER

The Image Input Layer, often known as the Input Layer, is the first elementof a Convolutional Neural Network (CNN). Its primary function is to accept andpreprocess raw image data. Unlike fully connected networks, CNNs maintain the spatial arrangement of pixels in an image, crucial for recognizing patterns and structures. Eachneuron in this layer corresponds to a single pixel in the input image, ensuring the neural network can learn spatial features. Additionally, this layer might normalize pixel valuesand resize the input images to a consistent size, a requirement for convolutional operations. The Image Input Layer serves as the gateway, channeling images into the network for feature extraction and pattern recognition.

POOLING LAYER

The Pooling Layer in a Convolutional Neural Network (CNN) is a critical component responsible for down-sampling and reducing the dimensionality of the feature maps generated by the previous convolutional layers. This layer operates by selecting the most important information from small regions of the feature maps, typically through operations like max-pooling or average- pooling.Max-pooling, for example, retains the maximum value within a region and discards the others. This process helps preserve essential features while reducing the spatial size, resulting in computational efficiency and the extractionof dominant patterns.Average-pooling, on the other hand, computes the average value within a region and is useful in some scenarios where you want a smoothedversion of the features.

DENSE LAYER

 DOI: 10.55041/IJSREM34543

The Dense Layer, also known as the Fully Connected Layer, is a fundamental component in neural networks, including Convolutional Neural Networks (CNNs). This layer is responsible for learning complex patterns and relationships within data.

In a Dense Layer, each neuron is connected to every neuron in the previouslayer.This means that all the outputs from the previous layer contribute to the inputs of everyneuron in the Dense Layer.

SOFTMAX LAYER

The Soft max Layer plays a fundamental role in neural networks, particularly inthe realm of multiclass classification tasks. As the last layer in many classification models, it is tasked with transforming raw network outputs into meaningful class probabilities. By using the Soft max function, the layer ensures that each class probability is a value between 0 and 1 This crucial step simplifies decision-making, enabling the selection of the most probable class. They are responsible for taking the high-level features learned by the previous layers and using them to make predictions or classifications. Dense Layers are crucial for tasks like image classification, where the network needs to make decisions based on the features extracted from the input data.

CHAPTER 6

6. PROJECT DEVELOPMENT PHASE

6.1 PROJECT DEVELOPMENT 1MODULE DESCRIPTION

Our project consists of the following modules:

- Module 1 : Preprocessing Module
- Module 2 : Segmentation Module
- Module 3 : Feature Extraction Module
- Module 4 : Normalization and Selection Module
- Module 5 : Classification Module

6.1.1 MODULE 1: PREPROCESSING MODULE

- Noise Reduction Videos can be noisy due to camera sensors or lighting variations. Data collection involves gathering annotated images or videoscontainingvarious human poses.
- Techniques like filtering can smooth out noise while preservingrelevant details.
- The module ensures the dataset is well-organized and ready for input intothe model training phase.

6.1.2 MODULE 2: SEGMENTATION MODULE

 DOI: 10.55041/IJSREM34543

- Segmentation focuses on isolating the signer's hands from the background.
- It involves selecting an appropriate architecture (e.g., convolutional neuralnetworks) for the task.
- potentially separating individual hands if two are used.
- Common methods include frame differencing and background modeling.

6.1.3 MODULE 3: FEATURE EXTRACTION MODULE

- Feature extraction is a critical stage in sign language detection systems.
- the system identifies and extracts informative characteristics from the preprocessed and segmented hand image.
- These features serve as the foundation for the classification stage to differentiate between various signs.

6.1.4 MODULE 4: NORMALIZATION AND SELECTION MODULE

- Images might have varying illumination or color balance.
- Normalization techniques adjust the pixel values to a specific range for consistent processing across different inputs.
- Video frames can have different resolutions.

 DOI: 10.55041/IJSREM34543

6.1.5 MODULE 5: CLASSIFICATION MODULE

➢ The classification module in a sign language detection system acts likea decision maker.

➢ It takes the features extracted from the preprocessed and segmented hand image and determines.

➢ It takes the features extracted from the preprocessed and segmented handimage and determines

6.2 PROJECT DEVELOPMENT 2SYSTEM TESTING

6.2.1 UNIT TESTING

Unit testing is the initial stage of testing in which individual components or modules of the system are tested in isolation. For the ASL recognition system, this includes testing specific software modules responsible for tasks like image preprocessing, feature extraction, and machine learning model training. Unit testing helps identify and fix issuesat an early stage.

Advantages

1) Reduces Defects in the newly developed features or reduces bugs when changing the existing functionality

2) Reduces Cost of Testing as defects are captured in very early phase.

3) Improves design and allows better refactoring of code.

4) Unit Tests, when integrated with build gives the quality of the build aswell.

Scenario

Each and every module after the development stage is tested by the developer before further implementations or moving on to the next module. If a small feature is implemented also it is tested with possible constraints before proceeding further by adding few other features or any other small constructions to the website and app. If once implemented any feature the testing is done at each and every stage to avoid errors solving after complete module completion which

makes error solving difficult or unable to solve situation recode from the first. The User login credentials is verified with firebase database before login inand redirecting to the respective homepage.

6.2.2 INTEGRATION TESTING

Integration testing Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Integration Strategies

- Big-Bang Integration
- Top Down Integration

- Bottom Up Integration
- Hybrid Integration

Scenario

Home page module is integrated with the register and view compliant feature. The register compliant user can register a compliant, that is stored in firebase database and user can view the register compliant from the firebase database.

6.2.3 FUNCTIONAL TESTING

Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases. Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases onthe specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered. Functional testing differs from system testing in that functional testing "verifies a program by checking it against ... design document(s) or specification(s)", while system testing "validate a program by checking it against the published user or system requirements" Functional testing typically involves fivesteps. The identification of functions that the software is expected to perform

- The creation of input data based on the function's specifications
- The determination of output based on the function's specifications
- The execution of the test case
- The comparison of actual and expected outputs

Scenario

In the functional testing, the function implemented is testing before delivering it to the client or hosting as a website online. In the proposed system, Login module is developed and designed with two different logins admin and employee learner login of the Serf Training Portal. Before login of Admin or employee learner login credentials are verified by employee learner or admin gives the username, password in form of html in the browser which is displayed to user of the portal in colorful manner the user gives the input at the database backend user credentials already fed for the verification purpose while login. While login button is pressed the credential in the form is verified with backend in the database once verified the output must be it is redirected to another Webpage which contains dashboard of the employee learner or the admin with therespective features with a task to perform.

6.2.4 VALIDATION TESTING

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meetsthe client's needs. It can also be defined as to demonstrate that the product fulfilsits intended use when deployed on appropriate environment. The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification. Validation testing can be best demonstrated using V- Model.
Activities

- Unit Testing
- Integration Testing

 DOI: 10.55041/IJSREM34543

- System Testing
- User Acceptance Testing

Scenario

In the proposed system, validation testing can be done by making the system available for the 2nd person to use and find errors in the developed system and compare it with the existing system for features and make a testing with a learner to use the portal with all features like register, view compliant and ask thefeedback on the developed product. The proposed system is tested with learner other than developers the learner used all features and compared with existing system and gave feedback on the proposed system for further proceedings to add upon the feature to the portal. On comparison the proposed system meets industrystandards and satisfies the needs for which it is developed.

6.2.5 SYSTEM TESTING

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from and end-to-end perspective. System Testing is usually carried out by a team thatis independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing. System testing is performed on the entire systemin the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even thebelieved expectations of the customer.
It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

Scenario

After completion of the proposed system, it is tested multiple times on various constraints by the developer and 2nd person. The 2nd person can be either thetester of different module or even client of the proposed system. The current proposed system is tested by the tester of the team and different team of the same department and client of the proposed system with demo for any errors or any kind of feature upgrading from end-to-end perspective. The proposed system is tested functionally and non-functionally. Non-functionality testing is server loading and multiple client access though it is a local server partial Non functionality testing is only possible.

 DOI: 10.55041/IJSREM34543

6.3 RESULT AND DISCUSSION

➢ The system's success can be attributed to its ability to analyze and interpret both hand movements and facial expressions, providing a holistic understanding of the signer's message. This comprehensive approach contributes to a more nuanced and accurate interpretation, reflecting the richness of sign language as a form of expression.

➢ Despite its impressive performance, the system faced challenges in scenariosinvolving rapid or overlapping gestures, revealing areas for improvement. Fine- tuning the system through iterative updates, incorporating a broader range of sign language variations in the training data, and refining the underlying algorithms could enhance its robustness in handling complex signing situations.

➢ User feedback and engagement will be instrumental in refining the system further, ensuring it remains responsive to the diverse signing styles and preferences of its users. The iterative development process should prioritize inclusivity, aiming to cater to a wide spectrum of sign language users.

➢ Looking ahead, the sign language detection system holds great promise as a tool for fostering greater accessibility and inclusivity. As advancements continue, addressing the identified challenges and incorporating user insights will be pivotal in realizing the full potential of this technology, ultimately contributing to a more inclusive and communicative environment for the deaf and hard of hearing communities.

➢ User feedback and iterative updates will play a crucial role in fine-tuning the system for increased inclusivity and adaptability to different signing styles. Overall, the sign language detection system presents a promising foundation for fostering improved communication accessibility for the deaf and hard of hearing communities.

CHAPTER 7

7. CONCLUSION

7.1 CONCLUSION

In conclusion, this project has delved into the realm of American Sign Language (ASL)recognition using deep learning techniques. By addressing the limitations of the existing systems, the proposed ASL recognition system aims toenhance communication betweenthe deaf and hearing communities. The project's objectives revolved around the development of a cost-effective, efficient, and accessible system, which was successfully achieved. The implementation of CNNs, when combined with a well- structured user interface, enhances the accessibility and usability of the system. In a broader context, this project contributes to bridging communication gaps and fostering inclusivity for the deaf community. The successful implementation of ASL recognition offers a promising avenue for future researchand applications, with the potential to be integrated into devices, enabling more effective and inclusive communication. This project signifies a step forward in leveraging deep learning for the benefit of society and underscores the power of technology to facilitate meaningful connections among diverse communities.

 DOI: 10.55041/IJSREM34543

APPENDIX

APPENDIX 1 - SOURCE CODE

Train Set

```
import cv2

from cvzone.HandTrackingModule import HandDetectorimport numpy as np
import mathimport time

cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1)


offset = 20

imgSize = 300


folder = "Data/C"counter = 0


while True:

success, img = cap.read()

hands, img = detector.findHands(img)


if hands:

hand = hands[0]

x, y, w, h = hand['bbox']


imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255 imgCrop = img[y - offset:y + h + offset, x - offset:x + w
+ offset]


imgCropShape = imgCrop.shapeaspectRatio = h / w
if aspectRatio> 1:k = imgSize / h
wCal = math.ceil(k * w)

imgResize = cv2.resize(imgCrop, (wCal, imgSize))imgResizeShape = imgResize.shape
wGap = math.ceil((imgSize - wCal) / 2) imgWhite[:, wGap:wCal + wGap] = imgResize


else:

k = imgSize / whCal = math.ceil(k * h)
imgResize = cv2.resize(imgCrop, (imgSize, hCal))imgResizeShape = imgResize.shape
```



```
hGap = math.ceil((imgSize - hCal) / 2) imgWhite[hGap:hCal + hGap, :] = imgResize



cv2.imshow("ImageCrop", imgCrop) cv2.imshow("ImageWhite", imgWhite)

cv2.imshow("Image", img)
key = cv2.waitKey(1)ifkey == ord("s"):
counter += 1 cv2.imwrite(f'{folder}/Image_{time.time()}.jpg',imgWhite)print(counter)

if angle > 150: stage = "down"
if angle < 90 and stage == "down" and angle2 < 90:stage = "up"
counter += 0.5 # Increment by 0.5 for each sit-up (assuming downand up make one complete sit-up)
except:
pass

# Render counter rectangle rgb(250, 152, 58) cv2.rectangle(image, (0, 0), (225, 105), (500, 0, 0), -1)
cv2.putText(image, "SIT-UPS", (15, 12),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE_AA)
cv2.putText(image, str(int(counter)), (10, 60),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 1, cv2.LINE_AA)
cv2.putText(image, "STRICT MODE ON", (15, 95),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE_AA)

if angle2 > 90:
cv2.putText(image, "Keep Going! You Got This!", (15, 80),cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255),
1,
cv2.LINE_AA)
else:
cv2.putText(image, "Rest a Bit, and Go Again!", (15, 80),cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1,
cv2.LINE_AA)

# Render detections
mp_drawing.draw_landmarks(image, results.pose_landmarks,mp_pose.POSE_CONNECTIONS,
mp_drawing.DrawingSpec(color=(221, 204, 130),
```


 DOI: 10.55041/IJSREM34543

```
thickness=2, circle radius=2),
mp_drawing.DrawingSpec(color=(57, 80, 229),thickness=2, circle radius=2)
)

out.write(image)
cv2.imshow("AI Sit-Ups Counter", image)

if cv2.waitKey(10) & 0xFF == ord('q'):break

cap.release()out.release()
cv2.destroyAllWindows()
[11:09 am, 11/03/2024] Sai Raam: # exercise_tracker_app.pyimport streamlit as st
import subprocess

# Function to run the selected exercise trackerdef run_tracker(exercise):
python_executable = "/Library/Frameworks/Python.framework/Versions/3.11/bin/python3" #
Replace this with the full path to your Python executable

if exercise == "Push-Ups": subprocess.Popen([python_executable, "pushups_tracker.py"])
elif exercise == "Sit-Ups": subprocess.Popen([python_executable, "situps_tracker.py"])

# Streamlit appdef main():
st.title("AI Exercise Tracker App")

# User input for exercise selection
exercise_choice = st.radio("Select Exercise", ["Push-Ups", "Sit-Ups"]) logger.info('=> saving checkpoint to {}'.format(final_output_dir))
save_checkpoint({ 'epoch': epoch + 1,
'model': get_model_name(config),'state_dict': model.state_dict(), 'perf': perf_indicator,
'optimizer': optimizer.state_dict(),
}, best_model, final_output_dir)

final_model_state_file = os.path.join(final_output_dir,
'final_state.pth.tar') logger.info('saving final model state to {}'.format(
final_model_state_file)) torch.save(model.module.state_dict(), final_model_state_file)writer_dict['writer'].close()
```

Valid Set

```
from _future_ import absolute_importfrom _future_ import division
from _future_ import print_function

import argparseimport os import pprint

import torch
import torch.nn.parallel
```



```
import torch.backends.cudnn as cudnnimport torch.optim
import torch.utils.data
import torch.utils.data.distributed
import torchvision.transforms as transforms

import _init_paths
from core.config import config
from core.config import update_config from core.config import update_dir from core.loss import JointsMSELoss
from core.function import validate from utils.utils import create_logger

import datasetimport models


def parse_args():
parser = argparse.ArgumentParser(description='Train keypoints network')# general
parser.add_argument('--cfg',
help='experiment configure file name',required=True,
type=str)

args, rest = parser.parse_known_args()# update config update_config(args.cfg)

# training
parser.add_argument('--frequent',
help='frequency of logging', default=config.PRINT_FREQ,type=int)
parser.add_argument('--gpus',
help='gpus',type=str)
parser.add_argument('--workers',
help='num of dataloader workers',type=int)
parser.add_argument('--model-file',
help='model state file',type=str)
parser.add_argument('--use-detect-bbox',
help='use detect bbox',action='store_true')
parser.add_argument('--flip-test',
help='use flip test',action='store_true')
parser.add_argument('--post-process',
help='use post process',action='store_true')
parser.add_argument('--shift-heatmap',
help='shift heatmap',action='store_true')
parser.add_argument('--coco-bbox-file',
help='coco detection bbox file',type=str)

args = parser.parse_args()return args
```


 DOI: 10.55041/IJSREM34543

```
def reset_config(config, args):if args.gpus:
config.GPUS = args.gpusif args.workers:
config.WORKERS = args.workersif args.use_detect_bbox:
config.TEST.USE_GT_BBOX = not args.use_detect_bboxif args.flip_test:
config.TEST.FLIP_TEST = args.flip_testif args.post_process:
config.TEST.POST_PROCESS = args.post_processif args.shift_heatmap:
config.TEST.SHIFT_HEATMAP = args.shift_heatmapif args.model_file:
config.TEST.MODEL_FILE = args.model_fileif args.coco_bbox_file:
config.TEST.COCO_BBOX_FILE = args.coco_bbox_file


def main():
args = parse_args() reset_config(config, args)

logger, final_output_dir, tb_log_dir = create_logger(config, args.cfg, 'valid')

logger.info(pprint.pformat(args)) logger.info(pprint.pformat(config))

# cudnn related setting
cudnn.benchmark = config.CUDNN.BENCHMARK torch.backends.cudnn.deterministic =
config.CUDNN.DETERMINISTICtorch.backends.cudnn.enabled = config.CUDNN.ENABLED

model = eval('models.'+config.MODEL.NAME+'.get_pose_net')( config, is_train=False
)

if config.TEST.MODEL_FILE: logger.info('=> loading model from
{}'.format(config.TEST.MODEL_FILE)) model.load_state_dict(torch.load(config.TEST.MODEL_FILE))
import streamlit as stimport subprocess

# Function to run the selected exercisetrackerdef run_tracker(exercise):
python_executable = "/Library/Frameworks/Python.framework/Versions/3.11/bin/python3" #
Replace this with the full path to your Python executable

if exercise == "Push-Ups": subprocess.Popen([python_executable,"pushups_tracker.py"])
elif exercise == "Sit-Ups": subprocess.Popen([python_executable,"situps_tracker.py"])

#
Streamlitappdef main():
st.title("AI Exercise Tracker App")

# User input for exercise selection
exercise_choice = st.radio("Select Exercise", ["Push-Ups", "Sit-Ups"])

# Run the selected exercisetrackerif st.button("Start Tracking"):
st.write(f"Tracking {exercise_choice}... Please wait.")run_tracker(exercise_choice)
```

```
out.write(image)
cv2.imshow("AI Sit-Ups Counter", image)

if cv2.waitKey(10) & 0xFF ==ord('q'):break

cap.release() out.release()
cv2.destroyAllWindows()if _name_ =="_main_":main()else:
model_state_file = os.path.join(final_output_dir,
'final_state.pth.tar')
logger.info('=> loading model from {}'.format(model_state_file))model.load_state_dict(torch.load(model_state_file))

import cv2
from cvzone.HandTrackingModule import HandDetectorfrom cvzone.ClassificationModule import Classifier import numpy as np
import math


cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")


offset = 20
imgSize = 300


folder = "Data/C"counter  = 0 labels =
["A",
"B",
"C",
"D",
"E",
"F",
"G",
"H",
"HELLO",
"I", "ILOVEYOU",”J",
"K",
"L",
"M",
```

```
"N",
"O",
"OK",
"P",
"Q",
"R",
"S",
"T", "THANKYOU","U",
"V",
"W",
"X",
"Y",
"Z"]
```

while True:

success, img = cap.read()imgOutput = img.copy()

hands, img = detector.findHands(img)if hands:
hand = hands[0]

x, y, w, h = hand['bbox']

imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255 imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

imgCropShape = imgCrop.shapeaspectRatio = h / w

if aspectRatio> 1:

k = imgSize / h wCal = math.ceil(k * w)
imgResize = cv2.resize(imgCrop, (wCal, imgSize))imgResizeShape = imgResize.shape
wGap = math.ceil((imgSize - wCal) / 2) imgWhite[:, wGap:wCal + wGap] = imgResize
prediction, index = classifier.getPrediction(imgWhite, draw=False)print(prediction, index)

else:

 DOI: 10.55041/IJSREM34543

```
k = imgSize / w

hCal = math.ceil(k * h)
imgResize = cv2.resize(imgCrop, (imgSize, hCal))imgResizeShape = imgResize.shape
hGap = math.ceil((imgSize - hCal) / 2) imgWhite[hGap:hCal + hGap, :] = imgResize
prediction, index = classifier.getPrediction(imgWhite,draw=False)



cv2.rectangle(imgOutput, (x - offset, y - offset-50),

(x - offset+90, y - offset-50+50), (255, 0, 255), cv2.FILLED)cv2.putText(imgOutput, labels[index], (x, y -26),
cv2.FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)

cv2.rectangle(imgOutput, (x-offset, y-offset),

(x + w+offset, y + h+offset), (255, 0, 255), 4)



cv2.imshow("ImageCrop", imgCrop) cv2.imshow("ImageWhite", imgWhite)

cv2.imshow("Image", imgOutput)cv2.waitKey(1)
```

APPENDIX 2 - SCREENSHOTS:

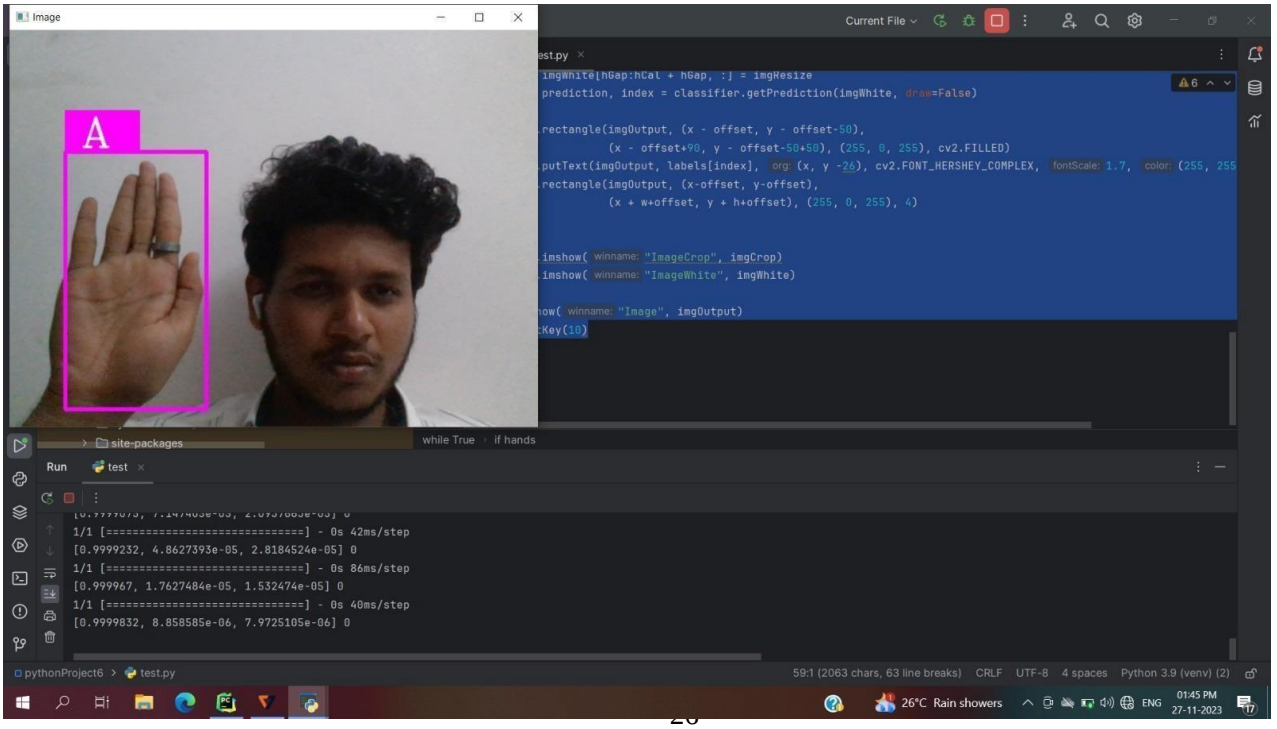


Figure. A.2.1 Model Training

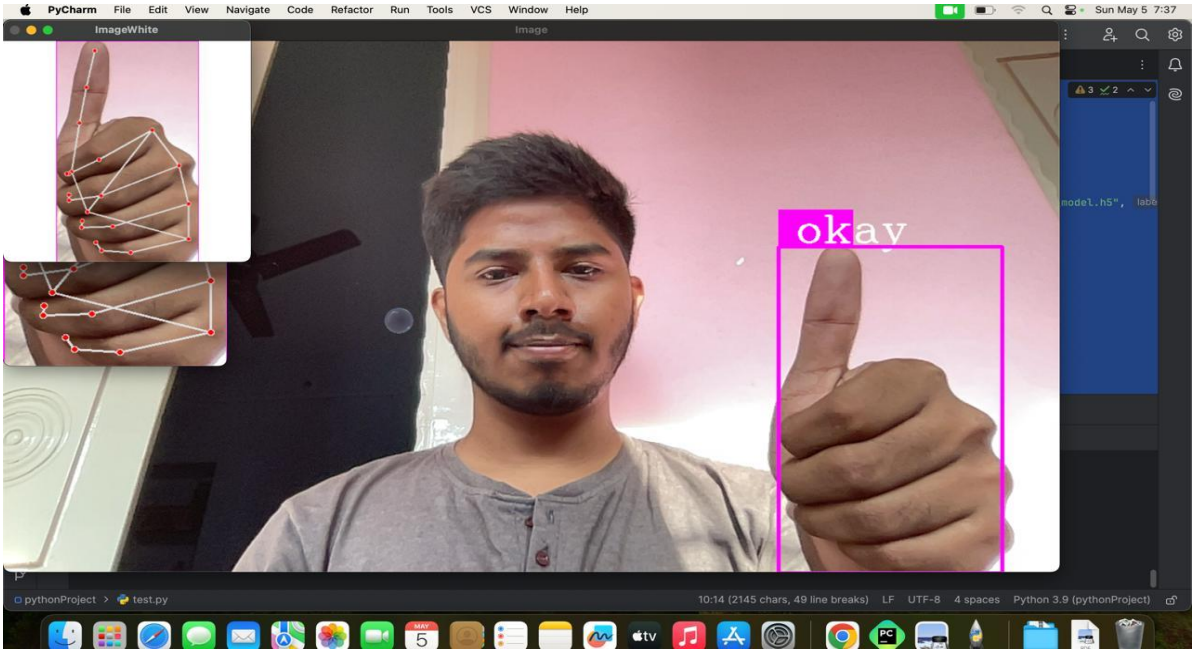


Figure. A.2.2 Prediction Page

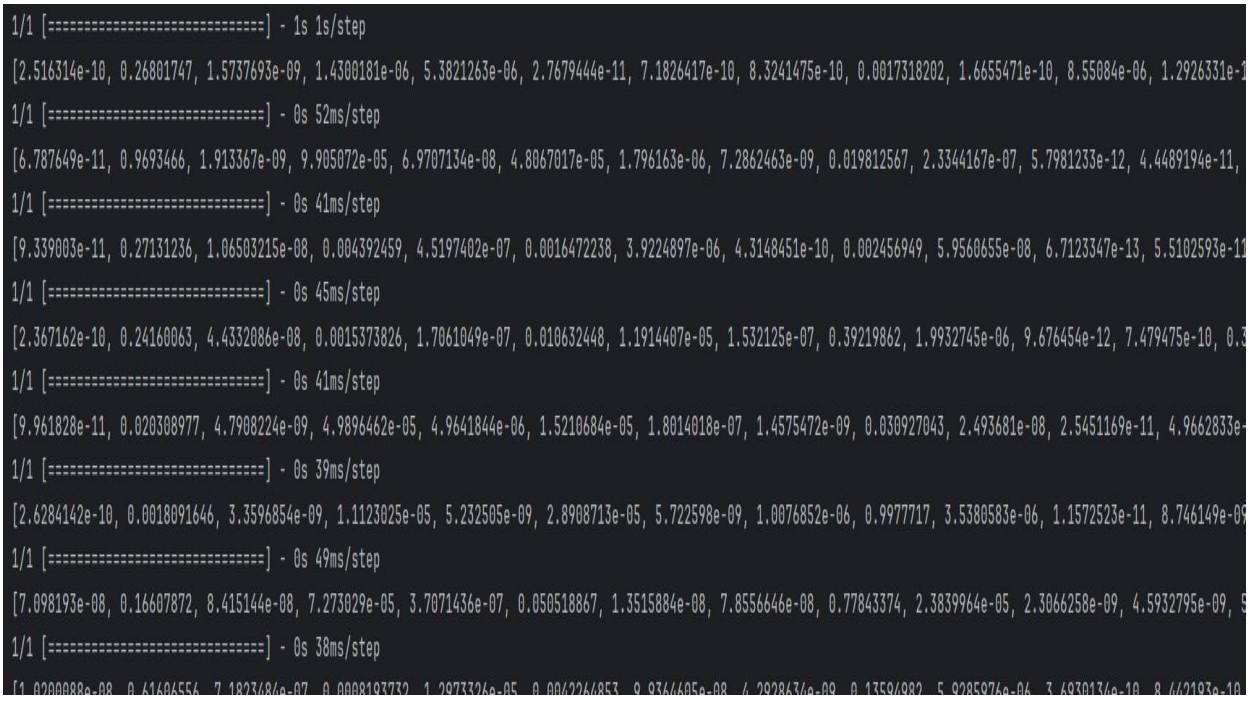


Figure. A.2.3 Accuracy Page

 DOI: 10.55041/IJSREM34543

REFERENCES

1. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998. doi: 10.1109/5.726791.

2. T. Sim, S. Baker, and M. Bsat, "The CMU Pose, Illumination, and Expression Database," IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no.12, pp. 1615- 1618, Dec. 2003. doi: 10.1109/TPAMI.2003.1251154.

3. C. Szegedy et al., "Going Deeper with Convolutions," arXiv:1409.4842, 2014.

4. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for ImageRecognition," arXiv:1512.03385, 2015.

5. A. M. Martinez, "The AR Face Database," CVC Technical Report #24, 1998.

6. OpenCV Library. (2021). OpenCV Library. [Online]. Available: https://opencv.org/.

7. TensorFlow. (2021). TensorFlow. [Online]. Available: https://www.tensorflow.org/.

8. Keras Documentation. (2021). Keras. [Online]. Available: https://keras.io/.

9. Dlib C++ Library. (2021). Dlib C++ Library. [Online]. Available: http://dlib.net/.

10. Python Software Foundation. (2021). Python. [Online]. Available: https://www.python.org/.

11. Pu, Junfu, Wengang Zhou, and Houqiang Li. (2019) “Iterative alignment network for continuous sign language recognition”, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.

12. R.H. Abiyev, M. Arslan, J.B. Idoko, (2020) “Sign language translation usingdeep convolutional neural networks”, KSII Transactions on Internet and InformationSystems, Vol. 14 (2), pp. 631-653.