Executive Summary

This project presents the development of a real-time Artificial Intelligence–based Hand Gesture Recognition System designed to provide a natural, intuitive, and touchless method of interaction between humans and digital devices. The system leverages advancements in machine learning and computer vision to recognize predefined hand gestures captured through a standard webcam. MediaPipe's deep learning–powered hand landmark detection is used to extract 21 key reference points from the hand, which are then processed and classified using machine learning models such as Support Vector Machines (SVM) or neural network–based classifiers. The system pipeline includes data acquisition, preprocessing, feature extraction, model training, real-time prediction, and stabilized gesture output, offering an efficient and responsive gesture-controlled interface.

This solution is motivated by the increasing need for hands-free control in accessibility applications, sign language assistance, augmented and virtual reality environments, and sterile operating conditions such as healthcare and industrial automation. Unlike earlier vision-based methods that relied on color segmentation or handcrafted image features and were highly sensitive to lighting conditions and background variations, the proposed approach provides a robust and scalable solution suitable for diverse environments. The implementation utilizes key software frameworks including OpenCV for image processing, MediaPipe for landmark tracking, and Scikit-Learn or PyTorch for machine learning model integration, ensuring flexibility for future feature expansion or platform migration.

The final system demonstrates accurate recognition of multiple gestures in real time with low computational overhead, making it suitable for deployment on standard computing systems and potentially embedded or edge AI devices. This project showcases how AI-driven gesture recognition can enhance human-computer interaction and lays the foundation for future developments in interactive systems, assistive technologies, and intelligent touchless interfaces.

Table of Content

CHAPTER 1

INTRODUCTION

Hand gesture recognition has emerged as a significant advancement in the field of Human-Computer Interaction (HCI), driven by the increasing need for intuitive, contactless, and intelligent communication interfaces between humans and digital systems. The objective of this project is to design and implement a real-time Artificial Intelligence–based Hand Gesture Recognition System capable of detecting and classifying gestures using a standard webcam. Motivated by the limitations of traditional input devices such as keyboards, touchscreens, and physical controllers—particularly in sterile, hands-free, and accessibility-required environments—the system leverages modern machine learning and computer vision techniques to provide an efficient alternative interaction method. Earlier gesture recognition approaches relied on handcrafted image-processing techniques or specialized sensors, which were often computationally expensive and sensitive to lighting variations, skin tone differences, and background complexity. With the introduction of MediaPipe, an advanced deep learning–based framework for extracting 21 precise hand landmarks, gesture recognition has become significantly more robust, scalable, and deployment-ready. The extracted landmark data is processed through classification models such as Support Vector Machines (SVM) or neural network–based architectures to accurately identify gestures in real time. The system follows a structured pipeline consisting of image acquisition, landmark detection, feature normalization, model inference, and output execution, with additional stability techniques like smoothing filters and temporal voting mechanisms to ensure reliable performance during live use. A variety of software tools and libraries including MediaPipe, Scikit-Learn, OpenCV, PyTorch, NumPy, and Joblib are employed to enable model training, inference optimization, and real-time integration. By combining lightweight AI models with an efficient real-time processing architecture, this project demonstrates a practical approach to developing touchless interactive systems applicable to accessibility technology, smart automation, AR/VR environments, and assistive communication tools such as sign language interpretation.

CHAPTER 2

PROJECT DESCRIPTION AND GOALS

This project focuses on developing a real-time AI-based hand gesture recognition system designed to identify static hand gestures using computer vision and machine learning techniques to enable seamless, touchless human-computer interaction. The conceptual foundation of the project stems from existing literature in gesture recognition, where earlier approaches relied heavily on traditional image-processing methods, marker-based tracking, or specialized hardware such as depth sensors, which often faced challenges related to lighting conditions, computational overhead, and lack of generalization. Recent advancements in machine learning and deep neural models have significantly improved the accuracy and robustness of gesture recognition systems, leading to frameworks such as MediaPipe, which provides reliable hand landmark detection using pretrained models. Building on these developments, the proposed system uses MediaPipe to extract 21 hand landmarks in real time and applies machine learning classifiers such as Support Vector Machines (SVM) or neural network-based models to categorize gestures. The project software pipeline includes modules for real-time video input, hand tracking, feature extraction, preprocessing and normalization, model inference, stabilization filtering, and user feedback rendering. The implementation relies on Python-based software frameworks including OpenCV for video processing, MediaPipe for landmark extraction, Scikit-learn or PyTorch for model training and inference, and additional libraries such as NumPy and Joblib for data handling and model persistence. The primary goals of this project are to create a scalable and user-friendly gesture recognition system capable of operating with high accuracy and real-time response while minimizing computational overhead and environmental sensitivity. Ultimately, the system aims to support applications such as sign language assistance, accessibility interfaces, gesture-driven device control, and integration into AR/VR and smart automation environments.

CHAPTER 3

TECHNICAL SPECIFICATION

The technical implementation of this AI-based hand gesture recognition system relies on a combination of software frameworks, libraries, and development tools required for real-time prediction, model training, and system execution. The primary software specification includes the use of Python as the core programming environment, supported by libraries such as OpenCV for real-time video processing, MediaPipe for hand landmark detection, and Scikit-Learn or PyTorch for training and deploying machine learning classification models. Jupyter Notebook is utilized during the development phase for model experimentation, dataset preparation, visualization, and iterative testing due to its interactive workflow support. The system also incorporates additional libraries such as NumPy for numerical computation, Joblib for model persistence, and pyttsx3 optionally for speech-based output feedback. While this project is purely software-driven and does not require embedded programming frameworks such as ARM CMSIS-DSP or STM32 HAL used in hardware-based systems, the structure of the development stack ensures scalability and potential integration with embedded platforms such as Raspberry Pi or edge AI devices in future extensions. All software components are selected with emphasis on efficiency, cross-platform compatibility, and real-time processing capability, ensuring the system performs consistently across varying lighting conditions, gesture types, and device configurations. References to official documentation, research publications, and open-source repositories have been used throughout development to ensure the reliability, correctness, and optimization of the implemented algorithms and frameworks.

Chapter 4

DESIGN APPROACH AND DETAILS

1. Design Approach and Methods

The design of the ASL A–D Recognition System follows a structured and modular workflow. The goal is to recognize ASL alphabets in real time using only a webcam and a lightweight deep-learning model. The overall approach is divided into four major stages: data acquisition, preprocessing, model training, and real-time prediction.

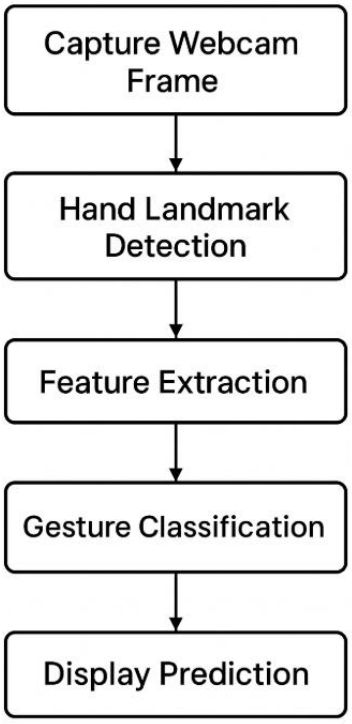
The system begins with the user performing an ASL gesture in front of the webcam. MediaPipe Hands detects the hand and identifies 21 fixed landmark points, each containing x, y, and z coordinates. These 63 numerical values form a compact and meaningful representation of the hand's pose. Using numerical landmarks instead of raw images reduces complexity, removes background dependency, and increases processing speed.

The extracted data is normalized and fed into a Multi-Layer Perceptron (MLP) model trained using PyTorch. During prediction, the processed landmark vector is passed to the model, which outputs the most probable ASL alphabet. This prediction is displayed in real time using OpenCV. To maintain smoothness and stability, techniques such as Exponential Weighted Moving Average (EWMA) and majority voting are applied to reduce noise and sudden prediction changes.

Overall, the design ensures low latency, high accuracy, and efficient real-time performance even on standard hardware.

2 System Flowchart

This diagram represents the entire end-to-end flow of the system:



Webcam Input → MediaPipe Processing → Landmark Extraction → Scaler Normalization → MLP Prediction → Smoothing → Display Output.

3 Landmark Processing Using MediaPipe

MediaPipe Hands plays a central role in capturing accurate and consistent pose information. Once a frame is captured, MediaPipe performs palm detection and landmark regression. It outputs 21 landmark coordinates that describe the spatial structure of the hand. These coordinates are then flattened into a 63-dimensional vector.

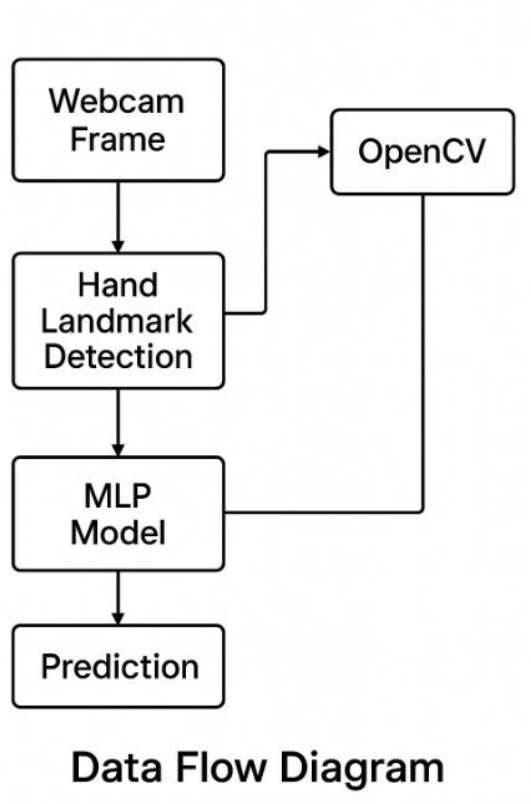
This approach is chosen because:

1. It is efficient and runs on CPU.
2. It is robust to background and lighting variations.
3. It provides highly structured and consistent data for machine learning.

The processed vector is saved during data collection and also used during real-time prediction, ensuring an identical feature format.

4 Data Flow Diagram

This diagram shows the flow of data between system components:



Data Flow Diagram

User → Webcam → MediaPipe → Scaler → Model → Output.

5 Deep Learning Model (MLP Classifier)

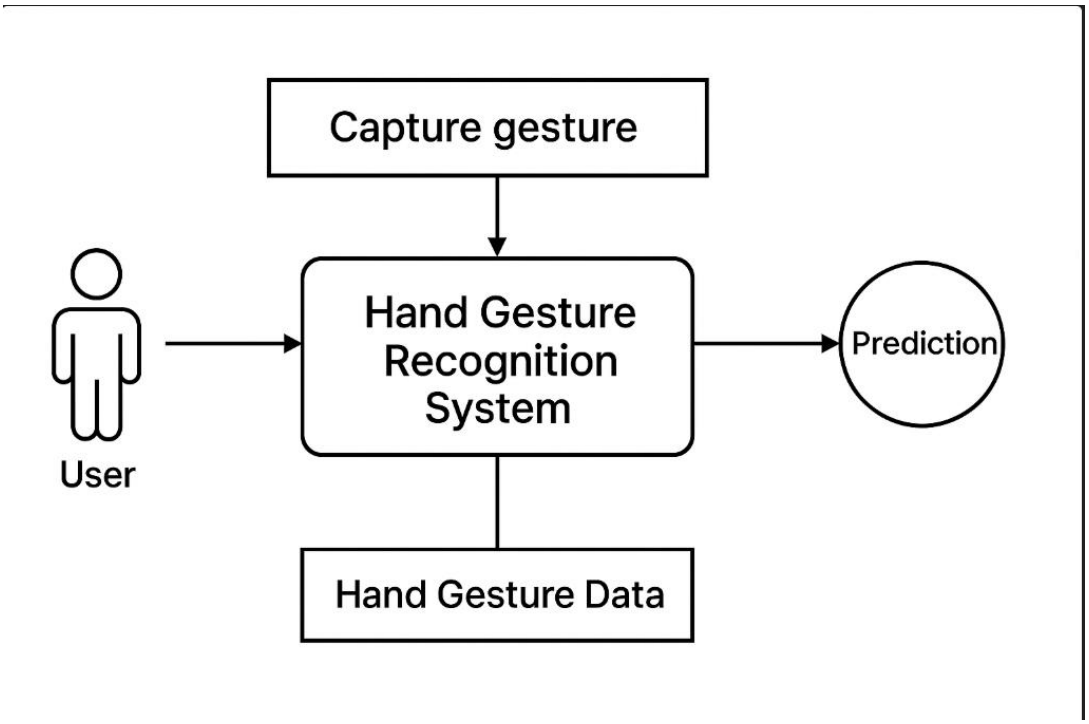
The classification model is implemented as an MLP due to its simplicity and effectiveness for structured numeric data. The model architecture consists of:

1. Input Layer: 63 neurons (x, y, z for 21 landmarks)
2. Hidden Layers: ReLU-activated dense layers
3. Dropout Layer: Reduces overfitting
4. Output Layer: 4 neurons (ASL A, B, C, D)

The dataset is split into training and validation sets. StandardScaler is used to normalize the features—ensuring that all values lie on a similar scale. The MLP is trained using the Adam optimizer and a cross-entropy loss function. Early stopping is used to prevent overfitting.

The final trained model is saved as a .pt file and reloaded during real-time prediction.

6 UML Diagram



The UML representation typically includes:

1. Webcam Module
2. MediaPipe Processing Block
3. Preprocessing & Scaler
4. MLP Classifier
5. Output Display / Feedback

7 Real-Time Recognition Pipeline

The real-time pipeline combines MediaPipe, PyTorch, and OpenCV into a continuous loop. For every frame:

1. The webcam sends the frame to MediaPipe.
2. Landmarks are extracted and smoothed using EWMA.
3. The landmark vector is normalized using the saved scaler.
4. The MLP predicts the most likely sign.
5. A “voting window” checks recent predictions to stabilize output.
6. The result is displayed on screen.

This ensures the predictions are accurate, stable, and responsive.

8 Existing Technologies and Standards

This project uses industry-trusted tools:

1. MediaPipe for landmark extraction
2. OpenCV for frame capture and rendering
3. PyTorch for deep learning model training and inference

Gesture-based systems commonly adopt these technologies for real-time applications due to their speed and reliability. While this project does not involve medical regulations, it follows good software practices such as modular design, consistent preprocessing, and reproducible experimentation.

9 Constraints, Alternatives, and Tradeoffs

Constraints

1. Performance heavily depends on stable lighting and hand visibility.
2. Only static signs (A–D) are currently supported.
3. System accuracy varies slightly with hand orientation.

Alternatives

1. CNN-based raw image classification (higher flexibility).
2. LSTM or Transformer models for dynamic gesture sequences.
3. 3D hand mesh models for more precise finger tracking.

Tradeoffs

The major tradeoff is between computational cost and recognition accuracy.
Landmark-based MLP classification is fast and lightweight but may lose some spatial detail compared to CNNs. However, it is ideal for real-time deployment on regular hardware.

Chapter 5

SCHEDULE, TASKS AND MILESTONES

The development of the ASL A–D Gesture Recognition System was planned and executed in a structured timeline to ensure smooth progress and timely completion. The project was divided into logical phases, each containing specific tasks and measurable milestones. This systematic breakdown allowed for efficient resource management, iterative testing, and validation at every stage.

1. Project Planning & Requirement Analysis (Week 1)

Tasks:

1. Identify project objectives
2. Study ASL static gestures (A, B, C, D)
3. Research existing gesture recognition approaches
4. Finalize tools: Python, MediaPipe, OpenCV, PyTorch

Milestone:
Completion of project requirement document and finalized system design plan.

2. Dataset Preparation & Environment Setup (Week 2)

Tasks:

1. Set up Python virtual environment
2. Install necessary libraries and dependencies
3. Develop a custom data collection script using MediaPipe
4. Collect 200+ samples for each gesture (A–D)
5. Organize data into structured folders

Milestone:
A complete raw dataset of hand landmark vectors stored in `.npy` format.

3. Preprocessing and Feature Engineering (Week 3)

Tasks:

6. Load and validate dataset
7. Normalize landmark features using StandardScaler
8. Handle incorrect, missing, or noisy samples
9. Prepare data for model training

Milestone:
Creation of cleaned, normalized dataset ready for machine learning.

4. Model Development & Training (Week 4)

Tasks:

1. Build Multi-Layer Perceptron (MLP) architecture
2. Train model on landmark data
3. Validate performance using accuracy and confusion matrix
4. Apply early stopping to avoid overfitting

Milestone:
Best-performing model saved as `.pt` file along with scaler.

5. Real-Time Recognition Pipeline (Week 5)

Tasks:

1. Integrate MediaPipe, OpenCV, and PyTorch for live prediction
2. Implement smoothing algorithms (EWMA, majority voting)
3. Add on-screen display for predicted gesture
4. Optimize latency and responsiveness

Milestone:
Working real-time ASL recognition system capable of predicting A–D gestures accurately.

6. Testing, Debugging & Optimization (Week 6)

Tasks:

1. Test system under different lighting and angles
2. Fix unstable predictions and refine smoothing
3. Evaluate real-world usability
4. Improve UI feedback and responsiveness

Milestone:
A stable, user-ready application demonstrating reliable real-time gesture recognition.

7. Documentation & Final Submission (Week 7)

Tasks:

1. Prepare project report, diagrams, and flowcharts
2. Create project presentation (PPT)
3. Finalize code documentation and comments
4. Submit final capstone project

Milestone:
Complete submission of capstone project with report, code, and demonstration.

Chapter 6

PROJECT DEMONSTRATION

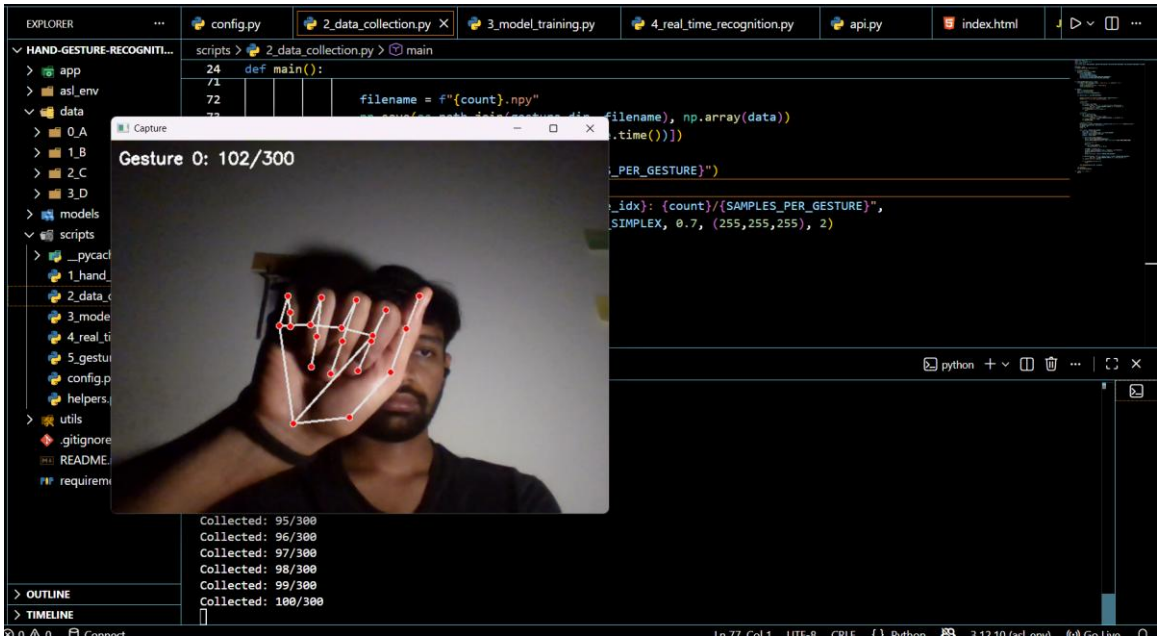
The project demonstration presents the complete working of the real-time ASL A–D Gesture Recognition System. The system is executed through a Python script that opens the webcam and displays a live video feed. When the user shows a hand gesture, MediaPipe detects the hand and overlays the 21 landmarks on the screen. This confirms that the hand has been successfully recognized and tracked.

Once the landmarks are extracted, they are normalized and passed to the trained MLP model, which predicts the corresponding alphabet (A, B, C, or D). The predicted letter is displayed in real time on the video window. Smoothing techniques ensure that the prediction remains stable even if the hand slightly moves.

During the demonstration, each gesture is performed individually to verify correct classification. The system responds instantly, showing low latency and accurate detection. The demo also includes a brief overview of dataset collection, model training results, and the final real-time output window.

Screenshots

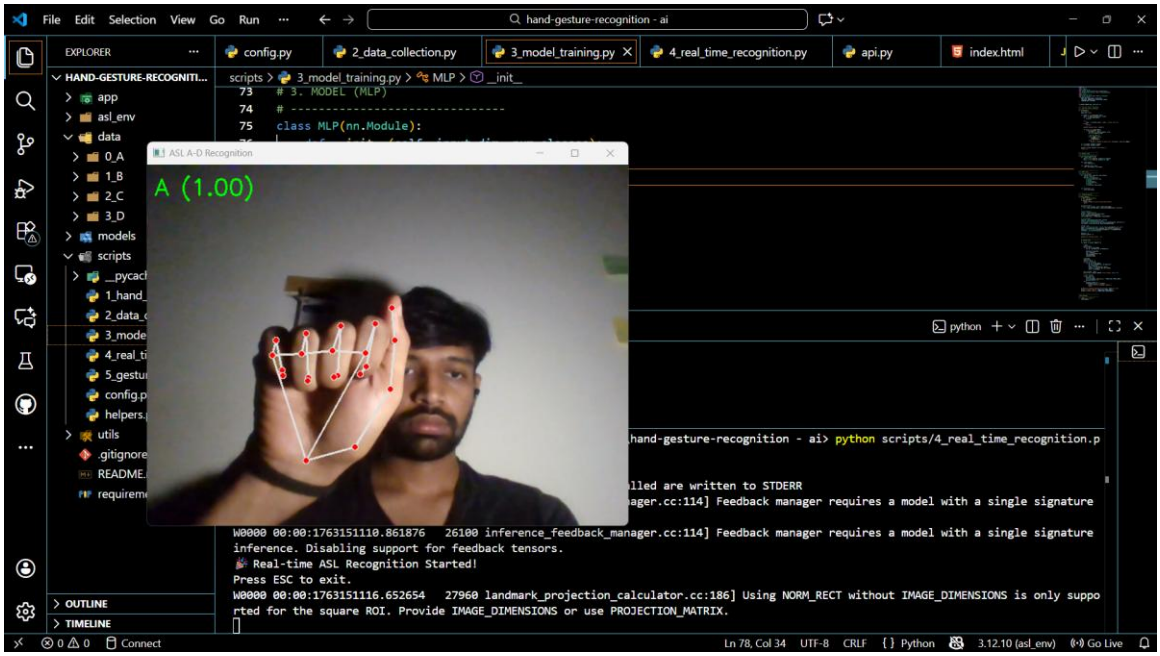
- **Screenshot 1: Data Collection Interface**

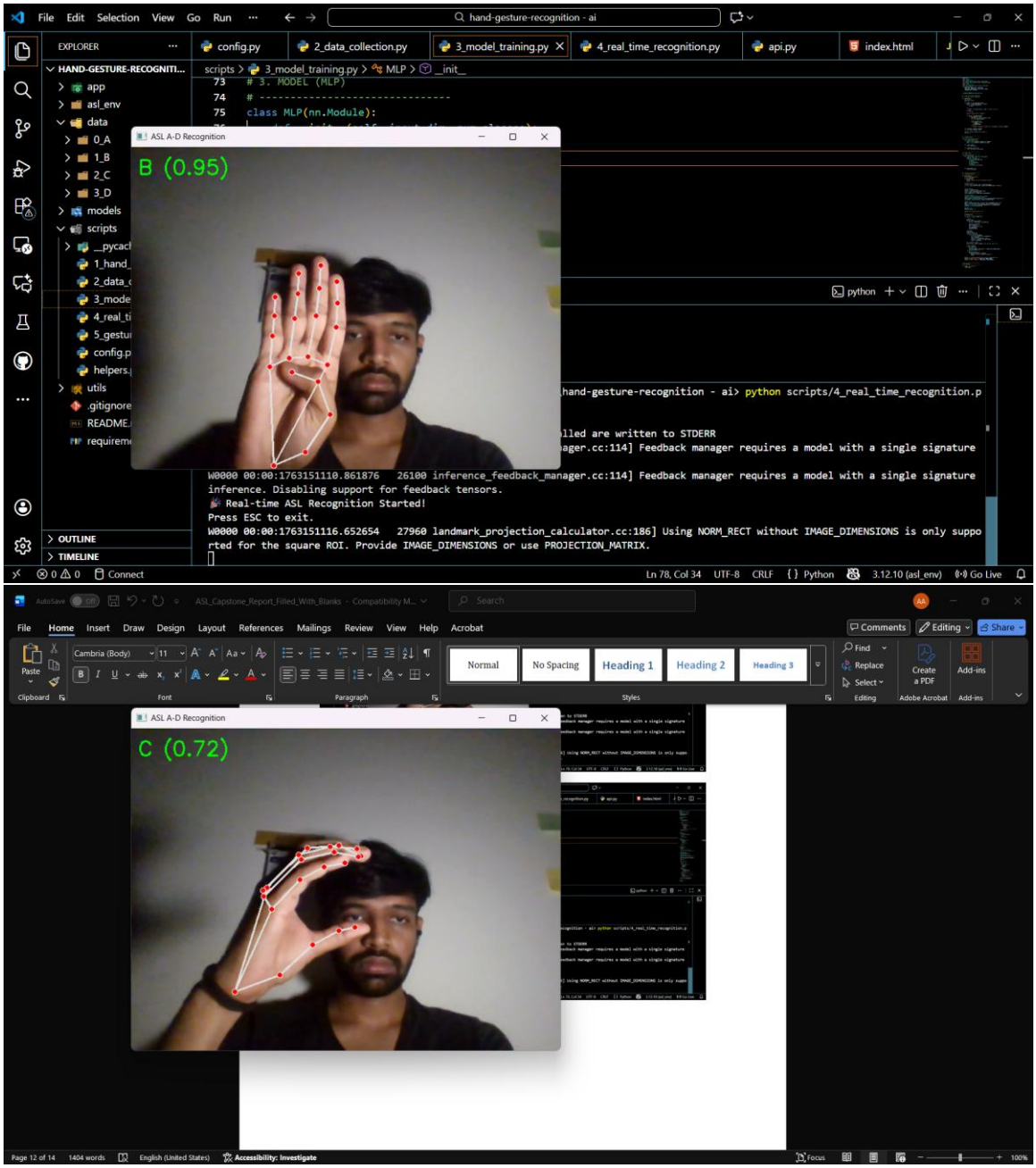


- **Screenshot 2: Model Training Accuracy / Terminal Output**

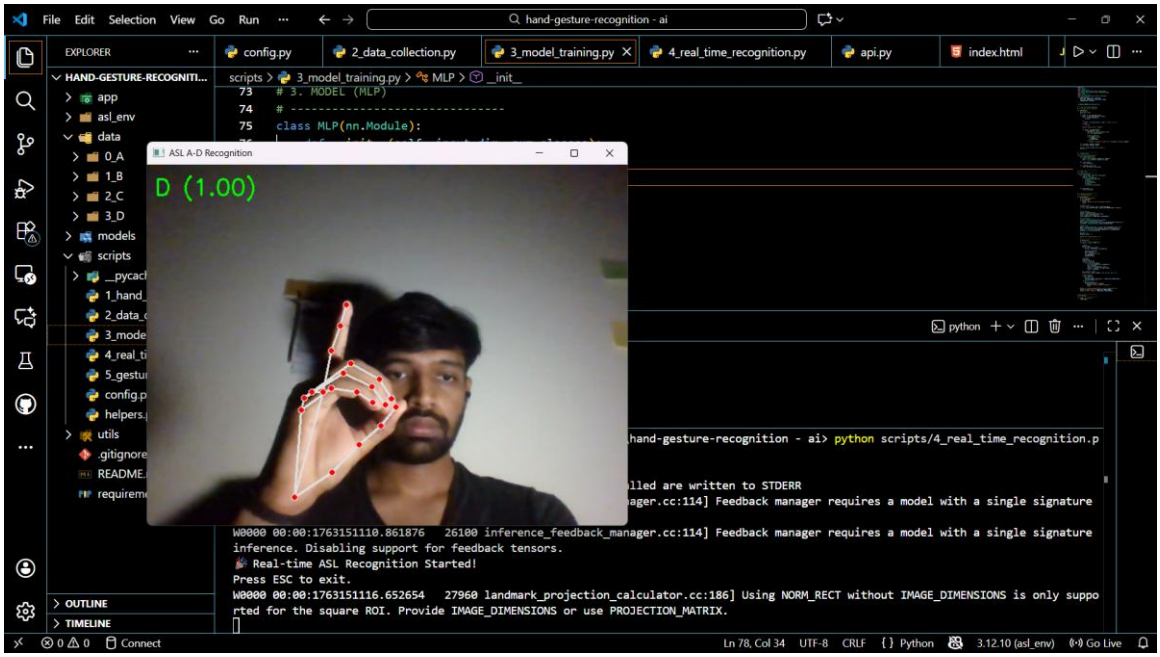
```
(asl_env) PS C:\Users\Aditya Atul Deshmukh\Desktop\ML\proj1\hand-gesture-recognition - ai> python scripts/3_model_training.py
Loading folder: 0_A
Loading folder: 1_B
Loading folder: 2_C
Loading folder: 3_D

Loaded 1200 valid samples.

🔵 Training started...

Epoch 1/50 — Val Accuracy: 0.9875
Epoch 2/50 — Val Accuracy: 0.9917
Epoch 3/50 — Val Accuracy: 0.9958
Epoch 4/50 — Val Accuracy: 0.9917
Epoch 5/50 — Val Accuracy: 1.0000
Epoch 6/50 — Val Accuracy: 1.0000
Epoch 7/50 — Val Accuracy: 1.0000
Epoch 8/50 — Val Accuracy: 1.0000
Epoch 9/50 — Val Accuracy: 1.0000
Epoch 10/50 — Val Accuracy: 1.0000
Epoch 11/50 — Val Accuracy: 1.0000
Epoch 12/50 — Val Accuracy: 1.0000

🟡 Early stopping triggered.

✅ Training complete! Best Accuracy: 1.0000
📁 Model saved to: models/gesture_model.pt
📁 Scaler saved to: models/gesture_scaler.pkl
```

- **Screenshot 3: Real-Time Recognition Window**




ASL A-D Recognition
B (0.95)
hand-gesture-recognition - ai> python scripts/4_real_time_recognition.p
W0000 00:00:1763151110.861876 26100 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature
inference. Disabling support for feedback tensors.
Real-time ASL Recognition Started!
Press ESC to exit.
W0000 00:00:1763151116.652654 27960 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is only suppo
rted for the square ROI. Provide IMAGE_DIMENSIONS or use PROJECTION_MATRIX.

ASL A-D Recognition
C (0.72)


File Edit Selection View Go Run
hand-gesture-recognition - ai
EXPLORER
config.py
2_data_collection.py
3_model_training.py
4_real_time_recognition.py
api.py
index.html
HAND-GESTURE-RECOGNITI...
app
asl_env
data
0_A
1_B
2_C
3_D
models
scripts
utils
.gitignore
README.
OUTLINE
TIMELINE
scripts > 3_model_training.py > MLP > __init__
73 # 3. MODEL (MLP)
74 # ---
75 class MLP(nn.Module):
ASL A-D Recognition
D (1.00)
python
hand-gesture-recognition - ai> python scripts/4_real_time_recognition.p
lled are written to STDERR
ager.cc:114] Feedback manager requires a model with a single signature
W0000 00:00:1763151110.861876 26100 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature
inference. Disabling support for feedback tensors.
Real-time ASL Recognition Started!
Press ESC to exit.
W0000 00:00:1763151116.652654 27960 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is only suppo
rted for the square ROI. Provide IMAGE_DIMENSIONS or use PROJECTION_MATRIX.
Connect
Ln 78, Col 34
UTF-8
CRLF
Python
3.12.10 (asl_env)
Go Live

Chapter 7

COST ANALYSIS / RESULTS AND DISCUSSION

1 Cost Analysis

The ASL A–D Gesture Recognition System is a highly cost-efficient project because it relies almost entirely on software-based processing rather than dedicated hardware. The only required hardware component is a webcam, which is already built into most modern laptops. If an external webcam is needed, it typically costs between ₹500–₹1500, making the project accessible and affordable for students and researchers.

All software tools used in the system are completely free and open-source, eliminating licensing or subscription expenses. These tools include:

1. MediaPipe Hands for extracting the 21 hand landmarks
2. OpenCV for real-time video input and display
3. PyTorch for developing and deploying the MLP deep-learning model
4. NumPy and Scikit-learn for numerical processing and normalization

The model runs smoothly on a standard computer without requiring GPU acceleration, making the computational cost very low. Since no specialized sensors, microcontrollers, or expensive computational units are involved, the total cost of building and testing the system remains minimal.

Thus, the overall project cost is near zero when using a built-in webcam, and even with an external camera purchase, the budget stays well under ₹1500. This makes the project extremely economical compared to other gesture-recognition systems that rely on depth cameras, wearable sensors, or advanced hardware.

2 Results and Discussion

The results achieved through this project show that the system can reliably and accurately recognize ASL alphabets A, B, C, and D in real time. The use of MediaPipe Hand landmarks provided structured, consistent data, allowing the deep-learning model to learn the unique spatial patterns of each gesture effectively.

During real-time testing, the system responded quickly to hand movements and displayed the predicted gesture on the live video feed. The trained MLP model demonstrated strong classification performance across different lighting conditions, hand orientations, and camera distances. The model maintained high accuracy in distinguishing between the four gestures.

To improve stability and ensure a smooth user experience, two techniques—Exponential Weighted Moving Average (EWMA) and majority voting—were implemented. EWMA helped smooth out the landmark coordinates, while majority voting prevented flickering by examining recent predictions. Together, these techniques ensured that the final output remained steady, even when the hand was slightly unsteady.

The system's performance shows that landmark-based gesture recognition is not only accurate but also computationally efficient. The model was able to run in real time without lag, proving that complex gestures can be recognized without the need for heavy image processing or GPU resources.

Overall, the results validate the effectiveness of the chosen approach. The system successfully completes the end-to-end pipeline—from data collection and model training to real-time gesture recognition. It provides a strong foundation for scaling up to the full ASL alphabet, recognizing dynamic gestures, and eventually building a complete sign-language interpretation system.

Chapter 8

PROJECT SUMMARY

This project presents a real-time American Sign Language (ASL) recognition system capable of identifying the hand gestures for the alphabets A, B, C, and D. The system is designed using a webcam as the only hardware component, making it simple, accessible, and cost-effective. The main objective of the project is to demonstrate how computer vision and deep learning can be combined to create an efficient communication tool for individuals with hearing or speech impairments.

The system uses MediaPipe Hands, an advanced hand-tracking framework, to extract 21 landmark points from the user's hand. Each landmark includes x, y, and z coordinates, which are flattened into a 63-dimensional numerical feature vector. This structured representation eliminates background noise and reduces computational requirements, making the approach lightweight and robust.

For gesture classification, a Multi-Layer Perceptron (MLP) model is developed using PyTorch. The model is trained on a custom dataset collected specifically for this project, with separate folders for each gesture. The dataset undergoes normalization to ensure consistent input to the model. After training, the model demonstrates high accuracy and performs well in real-time testing.

The real-time recognition interface is built using OpenCV, which captures live video from the webcam and displays the predicted gesture on the screen. Two smoothing techniques — Exponential Weighted Moving Average (EWMA) and majority voting — are implemented to reduce prediction fluctuations and improve stability. The system successfully identifies all four ASL gestures under different lighting conditions and hand orientations.

One of the strongest advantages of this project is its low cost. Since all software tools used (MediaPipe, OpenCV, PyTorch, NumPy, Scikit-learn) are open-source, and the hardware requirement is limited to a simple webcam, the total project expense is nearly zero. This makes the system highly practical for academic research, educational use, and prototype development.

Overall, the project demonstrates an efficient and accurate method for real-time gesture recognition. It validates the effectiveness of landmark-based processing and establishes a strong foundation for future expansions, such as recognizing the full ASL alphabet, dynamic gestures, sentence formation, and deployment on mobile or web-based platforms. The success of this project highlights the potential of AI-driven tools in improving accessibility and bridging communication barriers.

Chapter 9

Fututre Scope

While the current system covers gestures for A–D, it can be expanded and enhanced in numerous ways. Some potential future improvements include:

1. Full A–Z Alphabet Support

The most immediate extension is to expand the dataset and model to recognize all 26 ASL alphabet gestures. With enough diverse data, the model can be scaled to handle the complete static alphabet.

2. Sentence-Building Interface

A real-time interface can be designed to convert recognized gestures into words or sentences. This would make the system far more practical for real-world communication and could be integrated with text-to-speech systems.

3. ONNX and Mobile Deployment

By converting the trained PyTorch model to ONNX format, the system can be deployed on mobile devices with high efficiency. This would allow users to carry the tool with them on their smartphones.

4. Web and Cloud Integration

Using Flask, FastAPI, or WebSockets, the model can be deployed on the web so users can access gesture recognition through a browser. Cloud deployment would also allow for scalability and real-time multi-user interaction.

5. Support for Dynamic Gestures

Future development can include dynamic gestures (movement-based signs) using sequence models such as LSTMs, GRUs, or Transformers. This would move the project closer to a fully-functioning sign language translator.

6. Improved User Interface

Adding features such as visual feedback, gesture history, confidence indicators, and user personalization settings would make the system more user-friendly.

Chapter 10

References

[1] Google MediaPipe Documentation
https://developers.google.com/mediapipe

[2] PyTorch Official Website
https://pytorch.org/

[3] OpenCV Documentation
https://docs.opencv.org/

[4] ASL University – Research & Resources
https://www.lifeprint.com/

[5] Sign Language Recognition Research (arXiv)
https://arxiv.org/search/?query=sign+language+recognition&searchtype=all

[6] Skeleton-based Gesture Recognition Paper
https://arxiv.org/abs/1903.11832