REAL-TIME SIGN LANGUAGE RECOGNITION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

SHAFI S
TKM23MCA-2055

to

TKM College of Engineering

(Government Aided and Autonomous)

Affiliated to

The APJ Abdul Kalam Technological University

In partial fulfilment of the requirements for the award of the Degree of

MASTER OF COMPUTER APPLICATION

Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS
TKM COLLEGE OF ENGINEERING

November 2024

DEPARTMENT OF COMPUTER APPLICATIONS
TKM COLLEGE OF ENGINEERING
(Government Aided and Autonomous)
KOLLAM-691005

This is to certify that, this report entitled **REAL-TIME SIGN LANGUAGE RECOGNITION USING DEEP LEARNING** submitted by **SHAFI S** (**TKM23MCA-2055**), to TKM College of Engineering affiliated to APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the Degree of **Master of Computer Application** is a bonafide record of the project carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor(s)	**Mini Project CO-Ordinator**

DECLARATION

I undersigned hereby declare that the project report on **REAL-TIME SIGN LANGUAGE RECOGNITION USING DEEP LEARNING** , submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Application of the APJ Abdul Kalam Technological University, Kerala is a Bonafide work doneby me under supervision of **Dr.Fousia M Shamsudeen**. This submission represents my ideas in my own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. I also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data oridea or fact or source in our submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for theaward of any degree, diploma or similar title of any other University.

KOLLAM SHAFI S

11/11/24

ACKNOWLEDGEMENT

First and foremost, I thank GOD almighty and our parents for the success of this project. I owe sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project. I am extremely grateful to **Prof. Natheera Beevi M**, Head of the Department, Department of Computer Applications, for providing us with best facilities. I would like to thank my project guide **Dr.Fousia M Shamsudeen**. I would also like to express my sincere gratitude to my project coordinator, **Prof.Sheera Shamsu,** for her support and encouragement throughout this project.I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspiration throughout my course of study. I owe thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project.

SHAFI S

ABSTRACT

Sign language is an essential form of communication for individuals with hearing disabilities, but its limited comprehension by the broader public often impedes effective exchanges. This study explores the creation of an automated American Sign Language (ASL) recognition system through deep learning methodologies to close this communication divide. A tailored dataset consisting of 2000 instances of ASL gestures for each of the 26 letters was carefully gathered using OpenCV and a webcam, ensuring variability in lighting conditions and angles to improve model generalization. We analyze the performance of various leading deep learning models, such as a custom Convolutional Neural Network (CNN), VGG16, InceptionV3. Each model is evaluated based on its accuracy and response time during real-time gesture recognition, with the custom CNN crafted for lightweight and efficient predictions, while the pre-trained models leverage intricate architectures for better feature extraction. The goal of this research is to pinpoint the most effective model for real-time ASL recognition, offering significant insights into the development of automated sign language systems and enhancing human-computer interaction. This study aims to promote communication for the hearing impaired community and also helps to tackle the difficulties of real time gesture recognition by normal people.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Sign language (SL) is an essential parallel to all audible languages. It is considered the only language that connects vocal people with the hearing impaired community, which globally numbers around 430 billion with total deafness, and 1.5 billion are partially hearing impaired, according to the World Health Organization (WHO) . During the past few years, awareness of the importance of sign language has increased worldwide . All the hearing impaired-related associations motivate the countries to celebrate this community and empower them socially .

After the emergence of the computer and the possibility of processing natural languages, proposing an automated translation system for audible speech became most common over a range of generations . However, sign language has received less attention from scholars since it is structurally different from natural languages and requires more sophisticated techniques . Recently, Artificial Intelligence (AI) has slowly but steadily become a more significant part of our daily lives. The expanding usage of this technological revolution has exciting implications for the hearing impaired community . Subsequently, developing SL translation systems has become a realistic and applicable idea for researchers . Machine translation and image recognition have interested many researchers for several decades. The surveys indicate that initial research in this field was done in the 1940s . Recognizing sign language is deeply related to the machine translation field. instead, it can be considered a sub-domain with many exciting challenges to solve .

Sign language serves as an essential means of communication for those with hearing difficulties, enabling interaction and expression through visual signs. However, the widespread lack of proficiency in sign language can create substantial obstacles to meaningful communication, often resulting in social alienation and exclusion for many deaf people. Recent advancements in artificial intelligence (AI) and deep learning have opened up exciting possibilities for creating automated sign language recognition systems. These systems can serve as conduits, improving interaction between the deaf community and society at large. This study focuses on developing a robust, scalable, and real-time recognition system dedicated to American Sign Language (ASL).

To accomplish this, a comprehensive dataset of ASL signs was carefully assembled using OpenCV and a webcam, containing 2000 samples for each of the 26 letters in the alphabet. The dataset was recorded under various lighting conditions and angles to enhance the model's ability to generalize and perform effectively in different real-world scenarios.To achieve high accuracy and minimize latency in identifying ASL gestures, we investigate multiple advanced deep learning architectures, including a custom Convolutional Neural Network (CNN), VGG16, InceptionV3. Each model has distinct advantages in feature extraction and classification, enabling a comparative assessment of their effectiveness in recognizing hand signs. The custom CNN is tailored for lightweight and efficient real-time predictions, while the pre-trained models take advantage of deeper architectures and complex features for improved recognition.The main goal of this research is to reach a high degree of accuracy and real-time functionality, making the ASL recognition system useful for applications like translation services and assistive communication technologies. By analyzing and comparing the performance of different deep learning models on our ASL dataset, this study seeks not only to push forward the field of automated sign language recognition but also to enhance the broader area of human-computer interaction. Through this research, we aim to highlight the complexities of dataset creation, model efficiency, and the challenges faced in real-time gesture recognition, ultimately promoting greater inclusivity for the hearing-impaired population.

1.1 Existing System

Existing sign language recognition systems have evolved significantly over the years, driven by advancements in computer vision, machine learning, and deep learning. These systems aim to bridge the communication gap for deaf and hard-of-hearing individuals by translating hand gestures and signs into readable or spoken language.

Early systems primarily used traditional **computer vision** techniques, such as **image processing** and **pattern recognition**, to detect hand gestures. These systems worked by segmenting images, detecting edges, and extracting features like hand shapes or contours. However, they were often limited by environmental factors like lighting variations, complex backgrounds, and the diversity of hand shapes.

Additionally, systems using Google MediaPipe have become popular for real-time sign language recognition. MediaPipe provides efficient hand tracking by detecting and tracking 21 key landmarks on the hand in 3D space. This allows for the recognition of both static and dynamic gestures in real-time. The ability to track hands and fingers in real-time, along with the relatively low computational cost, makes MediaPipe-based systems highly effective for both research and commercial applications. Some systems use MediaPipe in combination with machine learning models, such as CNNs, to improve recognition performance and achieve high accuracy.
Despite these advancements, existing systems still face challenges. For instance, robustness to environmental variability (lighting, background noise, camera angle) remains an issue, and systems often need substantial computational resources to function in real-time. Moreover, datasets for sign language recognition remain limited, especially for less commonly used sign languages, and obtaining large, annotated datasets for training deep learning models continues to be a barrier for broader adoption.

.

1.2 Problem Statement

Sign language recognition systems have made significant advancements in recent years, primarily using Convolutional Neural Networks (CNNs) for hand gesture classification. However, several challenges still persist, hindering the development of robust and scalable systems. One of the key limitations is the scarcity of large annotated datasets, especially for less commonly studied sign languages, which severely limits the generalization of models across diverse real-world environments.

while CNN-based approaches have shown promising results, many existing studies fail to leverage the power of pretrained models, which have proven to be highly effective in transferring knowledge from large, publicly available datasets (such as ImageNet). Pretrained models like **VGG16**, **InceptionV3**, and others, have the advantage of being trained on vast amounts of diverse data, allowing them to extract rich feature representations and significantly reduce the need for large, domain-specific datasets. However, their potential remains underexplored in sign language recognition, where training models from scratch continues to be the norm.

Given the limitations in dataset size and the underutilization of pretrained models, there is a clear gap in research for developing a sign language recognition system that effectively handles the challenge of limited annotated data by collecting datasets manually and use pretrained models for feature extraction and also demonstrates improved accuracy and robustness, especially in real-world scenarios, by using pretrained CNNs such as **VGG16** and **InceptionV3**, which can help overcome dataset constraints and improve generalization across different sign languages and environmental conditions.

1.3 Proposed System

The proposed system for sign language recognition leverages a combination of manually collected datasets**,** pretrained deep learning models**,** and real-time processing to recognize American Sign Language alphabets (A-Z) and combine the predicted letters into words. The system is designed to address common challenges in sign language recognition, including limited dataset availability, variability in environmental conditions, and the need for robust and scalable models that can generalize well to diverse real-world scenarios.

The first step in the system is the **manual collection of data**. Using a webcam and OpenCV, sign language gestures representing the 26 letters of the alphabet (A-Z) are captured under different lighting conditions, hand shapes, and background settings. For each letter, 2,000 samples were collected, ensuring a diverse dataset that can better represent the variety of hand gestures seen in practice. The images are then preprocessed by normalizing pixel values, resizing them to fit the input size required by the models .

The system utilizes three different models: a **custom Convolutional Neural Network (CNN)**, **VGG16**, and **InceptionV3**, all trained on the manually collected dataset. The custom CNN is built specifically for hand gesture recognition, consisting of several convolutional layers followed by pooling layers and fully connected layers to classify the hand signs. This CNN is trained from scratch to learn features unique to the dataset. **VGG16** and **InceptionV3**, both pretrained on large image datasets like ImageNet, are used for transfer learning. These models are fine-tuned with the sign language dataset to adapt their learned features for hand gesture recognition.

Once the models are trained, their performance is evaluated using a test set, which was not part of thetraining data. Performance metrics such as accuracy, precision, recall, and F1-score are used to assess how well each model is able to recognize the letters in sign language. The results from the custom CNN, VGG16, and InceptionV3 are compared to determine the most effective model for recognizing hand gestures

After the individual letters are predicted by the model, the system moves to the next phase combining the predicted letters into words. This process involves tracking the sequence of predicted letters and grouping them into complete words. For real-time sign language recognition**,** the system integrates with OpenCV to capture video input from the webcam. Each frame of the video is processed by the trained models to predict the letter corresponding to the hand gesture in that frame. These predictions are then aggregated to form words and displayed on the user interface as text.

A graphical user interface (GUI) is developed to display the predictions in an easy-to-understand format. The predicted letters are shown as they are recognized, and the complete words are displayed after a sequence of letters is identified.

1.4 Objectives

- To develop a robust sign language recognition model that can accurately recognize hand gestures representing the **American Sign Language** alphabet (A-Z) from real-time video input. This system aims to bridge communication gaps for deaf and hard-of-hearing individuals by enabling effective interaction between sign language users and non-signers.
- To explore the possibility of pretrained models, specifically **VGG16** and **InceptionV3**, through transfer learning. By leveraging these models, the system benefits from the ability to generalize learned features from large, diverse datasets (such as ImageNet) to the task of sign language recognition, ultimately improving the model's accuracy.
- To develop a mechanism that can combine individual letter predictions into complete words. This will allow the system to not only recognize individual signs but also process continuous sign language gestures, enabling more fluent and meaningful communication. The model will track predicted letters in real-time and group them into words.
- To perform **real-time recognition**. This means that the system will be able to process live video input from a webcam and immediately display recognized letters and words, making it suitable for interactive applications such as communication aids and sign language learning tools.

CHAPTER 2
LITERATURE REVIEW

Sign language recognition has become an increasingly important field in assistive technology, aiming to bridge communication gaps for individuals who are deaf or hard of hearing. Traditional approaches to sign language recognition were primarily based on hand-crafted features and machine learning models, such as Support Vector Machines (SVM), k-Nearest Neighbor(k-NN). These methods relied on manual extraction of geometric and motion-based features, such as the shape, position, and movement of the hand. However, these approaches had limited success due to their dependence on extensive feature engineering and their inability to handle the complex, diverse nature of hand gestures in various real-world environments.

With the advent of deep learning, especially Convolutional Neural Networks (CNNs), sign language recognition systems have seen considerable improvements in accuracy and efficiency. CNNs are capable of automatically learning hierarchical features from raw image data, making them particularly effective for hand gesture recognition. Recent studies have demonstrated the effectiveness of CNN-based models for static gesture recognition. For instance, researchers have used CNNs to classify individual letters of the American Sign Language (ASL) alphabet with high accuracy, achieving 90%-99% recognition rates. However, CNN-based models often require large datasets to achieve optimal performance, which can be a significant limitation in the field.

2.1 Related Works

Recently, there has been an increasing interest in applying deep learning techniques for sign language recognition,which supports individuals with speech or hearing challenges. Numerous studies have been carried out on sign language recognition, employing different machine learning and deep learning technologies.

In their study , Ahmed Kasapbasi , Ahmed Eltayeb Ahmed Elbushra , Omar Al-Hardanee and Arif Yilmaz created a dataset along with a sign language interface system based on Convolutional Neural Networks (CNN) to translate gestures from sign language and hand shapes into natural language.

The neural network developed in this project is a CNN that improves the accuracy of recognizing the American Sign Language alphabet.This study introduces a novel dataset of the American Sign Language alphabet, considering various factors such as lighting conditions and distances They also compared their dataset with two other datasets from previous research.The proposed CNN model achieved an accuracy of 99.38% with outstanding prediction and a minimal loss of 0.0250.[1]

In a recent study,Sundar B and Bagyammal T introduced a vision-based system designed to recognize American Sign Language (ASL) alphabets by utilising Google's MediaPipe combined with Long Short-Term Memory (LSTM) networks. They created a custom dataset for the experimental study.The dataset encompassed 26 ASL alphabets, with every gesture being recorded over 30 frames, thereby improving the robustness of the model. Unlike conventional methods relying on geometric or shape-based features, the proposed system leverages MediaPipe's efficient 3D hand tracking to capture 21 hand landmarks from a single image frame, enhancing real-time accuracy.This method achieved an impressive 99% accuracy by taking advantage of MediaPipe's 3D hand landmark tracking, allowing for accurate,real-time gesture detection for both static and dynamic signs.[2]

The authors Prof. Mrs. Maheshwari Chitampalli, Dnyaneshwari Takalkar, Gaytri Pillai, Pradnya Gaykar and Sanya Khubchandani concentrates on creating a computer vision system intended for recognizing sign language gestures, with the goal of reducing communication barriers for individuals who are deaf or hard-of-hearing. The proposed system captures images of an individual signing through a camera, processes the visual frames to identify and interpret hand gestures, and converts them into text output .The main phases include data gathering, preprocessing, gesture segmentation, and feature extraction, with a Convolutional Neural Network (CNN) selected as the primary model for classification. The system underwent testing using an independent dataset to verify its accuracy, achieving an accuracy of 95%.This research highlights the significance of hand detection and tracking, feature extraction, and sign language translation, emphasizing the potential for automated sign language recognition systems to improve accessibility for non-signers and foster inclusive communication technologies.[3]

Hope Orovwode , Ibukun Deborah Oduntan and John Abubakar introduced a CNN-based approach for identifying static American Sign Language (ASL) alphabet signs, fulfilling a crucial demand for improved communication accessibility for people with hearing and speech disabilities. Using a dataset containing 44,654 images obtained through a HandDetector module, the research segmented the data into training, validation, and test sets, achieving outstanding accuracy in all phases. The model, designed with three convolutional layers along with a SoftMax output layer, was trained using the Adam optimizer and a categorical cross-entropy loss function, reaching a remarkable 99.86% training accuracy, 99.94% validation accuracy, and 94.68% test accuracy. This system surpassed earlier models, signifying an important advancement in the realm of sign language recognition, with the potential to close communication gaps for the Deaf and mute communities.OpenAI's GPT-3 the predecessor of GPT-4 introduced the capability of few-shot learning where the model can understand new tasks with minimal task-specific training examples. With 175 billion parameters GPT-3 achieved breakthrough performance in generating coherent human-like text setting the stage for advanced language understanding in conversational AI and creative writing applications [4].

Bader Alsharif , Easa Alalwany , Mohammad Ilyas proposed a highly accurate, real-time system for recognizing American Sign Language (ASL), intending to improve communication with the deaf community. By combining the YOLOv8 deep learning model with MediaPipe's hand landmark detection, the study created a strong classification system capable of identifying ASL gestures and converting them into text. MediaPipe identifies 21 key landmarks on the hand, which improves the YOLOv8 model's accuracy by capturing intricate hand positions. The authors trained the system using a large dataset of over 29,820 annotated ASL images, achieving an exceptional accuracy of 96.3% for the 26 letters of the alphabet, surpassing previous methods in recognition accuracy, class loss, and bounding box loss.[5]

Yulius Obi , Kent Samuel Claudio , Vetri Marvel Budiman , Said Achmad,, Aditya Kurniawan utilized the American Sign Language (ASL) Hand Sign Dataset,featuring 24 classes sourced from Kaggle with a Gaussian blur filter applied, and supplemented by additional classes from Nikhil Gupta to fulfill the 27-class requirement. This dataset consists of 30,526 images for training purposes and 8,958 images for testing, which is significantly more extensive than other datasets used for similar projects. The application reached an outstanding accuracy of 96.3%, underscoring the effectiveness of CNN-based systems for real-time sign language recognition and the advantages of larger, well-organized datasets inPublicly available datasets for sign language recognition are often small, leading to potential overfitting and reduced performance on unseen data. Additionally, existing datasets lack diversity, with limited variations in lighting, backgrounds, and user hand shapes. enhancing classification performance.[6]

From the previous researchs and works,the gap identified are ,

- There is a shortage in size of datasets available for sign language gestures for training the models.
- Most previous studies in sign language recognition have relied on custom CNN architectures without exploring the potential of advanced pretrained models like VGG16 and InceptionV3.
- Many studies focus on static images for each sign rather than dynamic, real-time gestures, which are closer to real-world applications.

CHAPTER 3

METHODOLOGY

3.1 Block Diagram

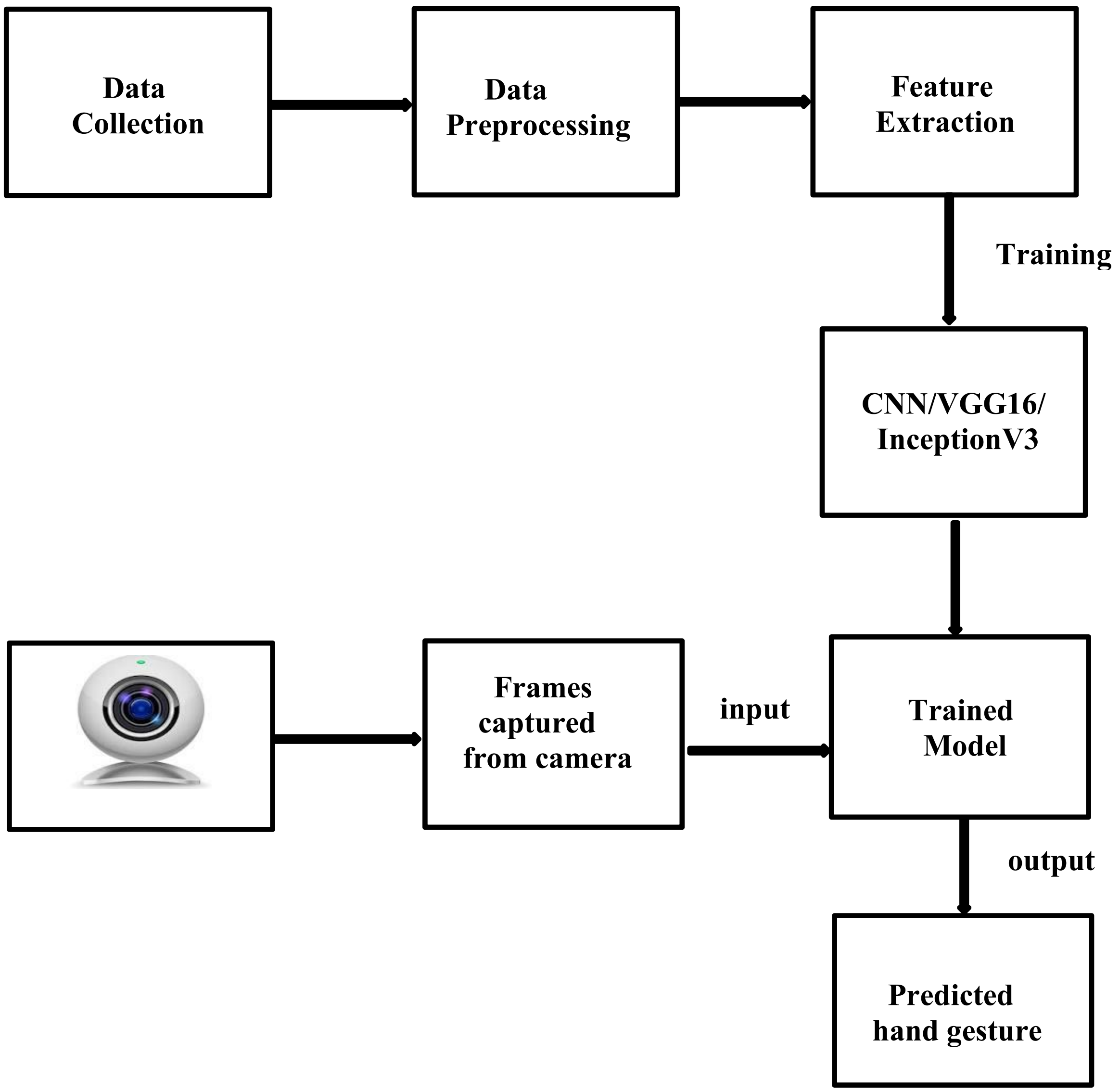


Fig 3.1: Block Diagram

3.2 Implementation

3.2.1 Dataset Collection

The dataset collection process involved manually capturing images of hand gestures for each of the 26 letters in the American Sign Language (ASL) alphabet, using a webcam and the OpenCV library. For each letter, 2,000 samples were gathered, resulting in a total of 52,000 images. To ensure diversity and improve model robustness, images were captured under varying conditions, such as different lighting, hand angles, backgrounds, and slight changes in hand shapes. This diversity in data aims to help the model generalize well across various real-world scenarios, enhancing its accuracy and reliability in recognizing ASL letters accurately in real-time applications.

3.2.2 Data Preprocessing

In the data preprocessing phase, the collected images were resized to match the input requirements of each model architecture—VGG16, the custom CNN, and InceptionV3. This resizing ensured that the images conformed to the specific input dimensions expected by each model, enabling effective training and feature extraction. Additionally, pixel values were normalized to standardize the dataset, enhancing the training stability and performance of the models. These preprocessing steps ensured that the data was properly formatted and optimized for each model, supporting consistent and accurate ASL gesture recognition.

3.2.3 Model Training

To achieve effective American Sign Language (ASL) recognition, we explored three different models: a custom Convolutional Neural Network (CNN), VGG16, and InceptionV3. Each model offers unique strengths that contribute to identifying hand gestures accurately. The custom CNN is specifically designed for this task, allowing it to learn unique ASL features from scratch. VGG16 and InceptionV3 are popular pretrained models known for their depth and complex architectures, which are fine-tuned on our ASL dataset to leverage their learned feature representations.

i. CNN

The Convolutional Neural Network (CNN) used in this project is a custom-designed architecture tailored specifically for American Sign Language (ASL) gesture recognition. CNNs are highly effective for image classification tasks as they can automatically and adaptively learn spatial hierarchies of features from input images through convolutional layers. This custom CNN architecture is composed of multiple convolutional layers with increasing filter sizes, each followed by pooling layers to reduce dimensionality and computational complexity. By learning relevant features of ASL hand gestures at various levels, this model aims to achieve high accuracy in distinguishing between the 26 letters of the ASL alphabet.

The CNN was trained using a sequential model structure in Keras, with an input size of 64×64×3 to match the preprocessed image dimensions. The architecture consists of four convolutional layers with ReLU activations, each followed by max-pooling layers to down-sample feature maps and Dropout layers to prevent overfitting. The network ends with a fully connected layer and a final output layer with 26 neurons, using a softmax activation for multi-class classification.

The training process involved 50 epochs with a batch size of 128, using an early stopping callback to monitor validation loss and halt training if performance did not improve for 5 consecutive epochs. The model was trained with train_generator for feeding images and evaluated with validation_generator for model performance assessment on unseen data, ensuring that the network generalized well to the ASL dataset.

ii. VGG16

VGG16 is a deep convolutional neural network architecture widely recognized for its success in image classification tasks. Developed by the Visual Geometry Group (VGG) at the University of Oxford, VGG16 consists of 16 weight layers, including 13 convolutional layers and 3 fully connected layers, making it deep enough to capture complex patterns in visual data. Known for its simplicity and depth, VGG16 uses small 3x3 filters across multiple layers, which helps the model learn detailed features while keeping computational costs manageable. For this project, VGG16 is used with transfer learning, leveraging its pretrained weights from large datasets to adapt to American Sign Language (ASL) gesture recognition.

To train VGG16 on the ASL dataset, we used transfer learning by loading the VGG16 base model with pretrained weights and adding custom layers on top. The base model layers were frozen, preserving the learned feature representations, while a fully connected and a dropout layer were added to enhance generalization. A final dense layer with softmax activation outputs predictions for each class in the ASL alphabet. The model was trained using a 80-20 split of the data, with 80% for training and 20% for validation. To optimize the model's performance, early stopping was implemented to halt training when validation loss did not improve for four consecutive epochs. Additionally, a model checkpoint callback saved the best weights based on validation accuracy, ensuring the optimal version of the model was retained. The training process was conducted over multiple epochs with a batch size suited for efficient training, using categorical cross-entropy as the loss function.

iii. InceptionV3

InceptionV3 is a sophisticated deep convolutional neural network model designed for high performance in image classification. Developed by Google, InceptionV3 is part of the Inception series, known for using parallel convolutional layers with varying filter sizes within "Inception modules." This unique structure allows the model to capture details at multiple scales in the input image, making it highly effective at extracting complex features. InceptionV3 consists of 48 layers and is pretrained on ImageNet, giving it robust feature representations that can be adapted for other image recognition tasks. In this project, we employ InceptionV3 for American Sign Language (ASL) recognition using transfer learning.

To adapt InceptionV3 for ASL recognition, we used the pretrained base model without its top layers, allowing it to learn specific features for ASL gestures. Additional layers were added on top of the base model, including a Global Average Pooling layer for reducing dimensionality, a fully connected layer and ReLU activation, and a dropout layer to prevent overfitting. A final dense layer with softmax activation provided class probabilities for each ASL letter.During training, all layers in the base model were frozen to retain the pretrained weights, while the newly added layers were trained on the ASL dataset. The model was compiled using the Adam optimizer and categorical cross-entropy as the loss function. Early stopping was implemented to monitor validation loss and prevent overfitting, with training stopping if validation loss did not improve over five consecutive epochs. A model checkpoint saved the best weights based on validation accuracy, ensuring that the highest-performing model was retained. This setup allowed InceptionV3 to achieve optimized performance in recognizing ASL gestures while minimizing computational costs.

3.2.4 Model Evaluation

Model evaluation is a critical step in assessing how well a trained model performs on unseen data. In this project, we use several metrics—accuracy, precision, recall, and F1-score—to gain a comprehensive understanding of each model's performance in recognizing American Sign Language (ASL) gestures. These metrics are particularly useful for classification tasks, as they provide insights not only into the model's general performance but also into its handling of specific classes. In evaluating machine learning models for American Sign Language (ASL) gesture recognition, it's essential to use a combination of metrics that provide insights into both overall performance and class-specific accuracy. For this purpose, we assess the model with four primary metrics: accuracy, precision, recall, and F1-score. Each of these metrics offers a unique perspective on model effectiveness, helping to understand how well the model generalizes to new data and how it handles specific classification challenges.

i. Accuracy

Accuracy is the ratio of correctly predicted instances to the total instances, giving an overall measure of model correctness. Accuracy is useful as an initial measure, providing an idea of how often the model correctly predicts the ASL gestures. However, it can be misleading if the dataset is imbalanced. For instance, if certain ASL letters appear more frequently than others, a model that performs well on these common letters but poorly on rare ones may still have high accuracy, masking weaknesses in identifying less frequent gestures.

Eqn 1: $$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}}$$

ii. Precision

Precision measures the accuracy of positive predictions by calculating the proportion of true positive predictions out of all positive predictions made by the model. Precision indicates how many of the model’s positive classifications (correct ASL gestures) are correct. High precision means the model has a low rate of false positives, which is critical in scenarios where incorrect predictions are costly or misleading. For example, in ASL recognition, high precision would ensure that each predicted gesture is highly likely to be correct, providing confidence in the model's outputs.

Eqn 2: $$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

iii. Recall

Recall (or sensitivity) measures the model's ability to correctly identify all instances of a particular class by calculating the proportion of true positives out of all actual positives. High recall indicates that the model successfully captures most instances of each ASL gesture, meaning fewer instances are missed. In applications like ASL recognition, where capturing all relevant gestures is important, high recall ensures that the model is not overlooking potential instances of a gesture. However, focusing too much on recall without considering precision could lead to more false positives.

Eqn 3: $$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

iv. F1-Score

F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. The F1-score is valuable when there is a trade-off between precision and recall, particularly if the dataset has an imbalance. A high F1-score indicates that the model has achieved a good balance, effectively identifying true positives while avoiding false positives and false negatives. In ASL recognition, a high F1-score means the model accurately identifies gestures and minimizes incorrect predictions, making it suitable for real-time applications.

Eqn 4 : $$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

3.2.5 Model Selection

Model selection is the process of choosing the best model for a specific task from a set of candidate models. For our American Sign Language (ASL) recognition project, the goal is to select a model that balances high accuracy with efficient real-time performance, enabling reliable ASL gesture recognition. We trained and evaluated multiple models, including a custom Convolutional Neural Network (CNN), VGG16, and InceptionV3, each with unique architectures and strengths. Our selection process involves comparing their performance based on evaluation metrics, computational efficiency, and the specific requirements of ASL recognition.

Each model's effectiveness was assessed using key evaluation metrics—accuracy, precision, recall, and F1-score. High accuracy is essential to ensure correct predictions, while high precision and recall indicate the model's reliability in identifying ASL gestures. The F1-score helps us understand the balance between precision and recall, which is particularly important when certain gestures may be more challenging to classify.

A critical factor in model selection is generalization, or the model's ability to perform well on unseen data. We test the models on a separate validation set that was not used during training to assess their generalization. A model with high generalization capability is less likely to overfit, meaning it can better handle variations in hand shapes, lighting, and backgrounds in real-world scenarios.

Since real-time ASL recognition is a primary goal, computational efficiency is a key consideration. Models like InceptionV3, with a complex architecture, may have high accuracy but can be computationally intensive, which might slow down real-time processing. In contrast, a custom CNN is often lighter and faster, which may be beneficial for real-time applications.

After evaluating each model based on the above criteria, the selected model is the one that best balances high accuracy and computational efficiency, making it well-suited for real-time ASL gesture recognition. For instance, if the custom CNN achieves comparable accuracy with a lighter architecture, it may be chosen over more complex models. Alternatively, if VGG16 or InceptionV3 demonstrates superior performance in both accuracy and generalization, one of these models might be chosen despite the additional computational requirements.

3.2.6 Model Deployment

For deploying the American Sign Language (ASL) recognition model, we are building a user-friendly desktop interface using customtkinter, a modern framework for developing visually appealing and responsive interfaces in Python. The desktop interface will allow users to interact with the ASL recognition system in real time, making it accessible and intuitive. Once the trained model (custom CNN, VGG16, or InceptionV3) is loaded into the interface, it will be ready to process live video input

from the user's webcam. The model will analyze each frame in real time, recognize the displayed ASL gesture, and display the predicted letter on the interface. The predicted letters will then be combined into words, providing continuous feedback as the user performs ASL gestures. Using customtkinter ensures that the interface is not only functional but also visually cohesive, enhancing the user experience while maintaining high performance and efficient model inference. This setup enables a practical, real-time ASL recognition solution that can run smoothly on a desktop environment.

3.3 Software Requirements

3.3.1 Operating System

The American Sign Language (ASL) recognition system is designed to be cross-platform, meaning it can run on various operating systems that support Python and the necessary libraries. The operating system plays a crucial role in providing the underlying platform for all components of the system to work together efficiently.

On **Windows 10 or 11**, the system is fully compatible, as Python, TensorFlow, OpenCV, and customtkinter all have stable support on this platform. Python can be easily installed via the Microsoft Store or downloaded directly from Python's official website. Libraries and dependencies can be managed using pip or a virtual environment such as Anaconda. Additionally, Windows supports real-time webcam access via OpenCV, making it suitable for the real-time prediction aspect of the project.

For **Linux**, particularly distributions like Ubuntu, the system is highly compatible and well-suited for machine learning projects due to the efficient resource management and robust support for open-source tools. Python and required libraries can be installed using package managers like apt or yum, or via pip in virtual environments. Linux provides better memory and CPU resource management, which is beneficial for training models and real-time prediction, especially on machines with limited resources.

On **macOS**, the system is also compatible with Python and the necessary libraries, although it doesn't currently support CUDA for GPU acceleration. However, macOS devices, particularly those with Apple Silicon (M1/M2) processors, can still run the system efficiently. The latest versions of TensorFlow are optimized for Apple Silicon, providing enhanced performance for model inference, even without GPU support. Python can be installed via the official installer or using Homebrew, and the necessary libraries can be managed through pip or within a virtual environment.

3.3.2 Development Tools

i. Visual Studio Code IDE

Visual Studio Code (VS Code) is a lightweight, open-source code editor developed by Microsoft. It is widely used by developers due to its flexibility, powerful features, and support for multiple programming languages, including Python, JavaScript, C++, and many more. VS Code is designed to be simple and fast while providing a wide range of features that enhance the development experience. One of the key features of VS Code is its **extensibility**. It supports a rich ecosystem of extensions that can be added to improve functionality. For Python development, the Python extension provides features such as IntelliSense (code completion and suggestions), debugging tools, linting (real-time code quality checks), and unit testing integration. This makes it an excellent choice for Python developers, particularly those working on machine learning and AI projects, like the ASL recognition system.VS Code also has an integrated **terminal**, allowing developers to run commands, scripts, and interact with version control systems (such as Git) directly from the editor, without needing to switch between applications. This feature helps streamline the development workflow by keeping all the necessary tools in one interface.

ii. Git

Git is a distributed version control system that enables multiple developers to collaborate on software projects by tracking and managing changes to code over time. It allows users to maintain different versions of their code, providing a robust mechanism for branching, merging, and tracking changes in a project. Git enables seamless collaboration by allowing multiple developers to work on the same codebase without overwriting each other's changes. Each developer can clone a repository, make changes locally, and then push their changes to a shared remote repository, where others can pull and review them. Git also allows for easy rollbacks, ensuring that developers can revert to previous versions of their code if necessary. This makes it particularly useful for managing complex projects, including those that involve large datasets and models, like your sign language recognition system, where multiple iterations and experiments may be required. With platforms like GitHub, GitLab, and Bitbucket, Git has become a standard tool for collaborative software development, providing version control, backup, and project management features.

iii. Google Colab

Google Colab is a cloud-based platform that provides an interactive environment for writing and executing Python code, making it ideal for data science and machine learning tasks. It offers free access to powerful hardware accelerators like GPUs and TPUs, which significantly speed up the training process

for deep learning models, such as CNNs, VGG16, and InceptionV3. By leveraging these resources, users can train large models on complex datasets, like the one used for sign language recognition, much faster than on local machines. Colab eliminates the need for local setup, as it comes with pre-installed libraries and allows easy installation of additional packages. With seamless integration to Google Drive, users can store and manage their data and model files in the cloud, ensuring easy access and collaboration. Colab’s interactive interface also supports real-time code execution and visualization, making it easier to track model performance and iterate on designs without worrying about system configurations or hardware limitations.

3.3.3 Programming Language and Libraries used

i. Programming language - Python

Python is a high-level, interpreted programming language known for its simplicity and readability, making it an excellent choice for both beginners and experienced developers. Its clean and easy-to-understand syntax emphasizes readability, which reduces the cost of program maintenance. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It is dynamically typed, meaning that you do not need to declare the data types of variables explicitly, and it features automatic memory management through garbage collection.

Python plays a crucial role as the primary programming language due to its simplicity and the extensive libraries available for machine learning, data processing, and image manipulation. Python's clean and readable syntax makes it easy to implement and iterate on complex algorithms, such as the deep learning models you're using (CNN, VGG16, and InceptionV3) for hand gesture recognition. The language supports popular machine learning frameworks like TensorFlow, Keras, and PyTorch, which are essential for training and fine-tuning the models on manually collected dataset. Additionally, Python's OpenCV library enables real-time video capture and image processing, which is vital for implementing the webcam-based sign language recognition feature in the project.

ii. Tensorflow

TensorFlow is an open-source machine learning framework developed by Google that enables users to build, train, and deploy machine learning models efficiently. It is designed to handle the complexities of large-scale machine learning tasks, offering a comprehensive ecosystem for building and managing deep learning models. TensorFlow's core strength lies in its ability to perform fast and efficient mathematical computations, making it ideal for tasks such as training deep neural networks, handling large datasets, and supporting both CPU and GPU computations.

In addition to its lower-level capabilities, TensorFlow also provides high-level APIs, such as Keras, which simplifies the process of building and training deep learning models. Keras offers an intuitive and easy-to-use interface for designing neural networks, making TensorFlow accessible for both beginners and experienced developers. It supports a wide range of neural network architectures,

including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and models for transfer learning, making it suitable for various tasks like image classification, object detection, and natural language processing.

TensorFlow's high-level API, Keras, simplifies the process of designing and training these models. Using Keras, I can quickly define the architecture of the CNN or modify pretrained models like VGG16 and InceptionV3 to better recognize the hand gestures in my dataset. TensorFlow's ability to perform fast computations using GPUs accelerates the training process, allowing us to efficiently process the 2000 samples per alphabet letter collected for the sign language dataset. This is crucial when working with large datasets and deep learning models, as it helps reduce the training time and improve the performance of the models.

iii. Keras

Keras is a high-level neural networks API that runs on top of TensorFlow, designed to simplify the process of building and training deep learning models. It provides a user-friendly interface that allows developers to quickly define complex models without dealing with the low-level operations that are required in traditional machine learning frameworks. Keras is particularly valuable in projects like sign language recognition, where rapid prototyping and experimentation are essential. By using Keras, I can easily design, train, and fine-tune models like Convolutional Neural Networks (CNNs), VGG16, and InceptionV3, which are the core architectures used in our sign language recognition system.In the context of our project, Keras simplifies the process of creating and modifying models. For example, when training the custom CNN, I can easily stack layers such as convolutional layers, pooling layers, and dense layers to build the model architecture. Keras also makes it simple to apply regularization techniques, like Dropout, which helps prevent overfitting in the models. For pretrained models like VGG16 and InceptionV3, Keras enables easy integration of transfer learning, where I can fine-tune these models on the sign language dataset to adapt their features to recognize hand gestures more effectively.Keras is built to work seamlessly with TensorFlow, leveraging its power to train models efficiently using GPUs. It supports a wide range of machine learning tasks, including image classification, which is the primary function in this project.

iv. NumPy

NumPy is a powerful, open-source library in Python that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It is a fundamental package for scientific computing and is widely used in machine learning, data analysis, and computer vision tasks. In the context of our sign language recognition project, NumPy plays a critical role in handling and processing the image data, performing mathematical operations, and ensuring efficient data manipulation throughout the machine learning pipeline.

In this project, NumPy is essential for working with the large datasets of hand gesture images. When processing images for input into models like CNN, VGG16, or InceptionV3, NumPy is used to store the image data as arrays. These arrays are the numerical representation of images, where each pixel is converted into a numerical value (usually an integer or float). NumPy allows us to efficiently perform operations such as resizing the images, normalizing pixel values, and reshaping the data to fit the input dimensions required by the models.

Additionally, NumPy provides many functions that are useful during model training and evaluation. For example, when splitting the dataset into training and testing sets, NumPy can be used to handle the array operations required for splitting and shuffling the data. It also facilitates operations like data augmentation, where images can be manipulated (e.g., rotated, flipped, or zoomed) by applying mathematical transformations to the image arrays. NumPy's efficiency with array operations helps ensure that the model training process is optimized, especially when working with large-scale datasets.

v. Pandas

Pandas is a widely-used open-source library in Python that provides powerful tools for data manipulation and analysis. It offers two primary data structures: **Series** (1-dimensional) and **DataFrame** (2-dimensional), which are highly efficient for handling and analyzing structured data. Pandas is built on top of NumPy and is designed to work seamlessly with numerical data, making it an essential tool for tasks such as data cleaning, transformation, and exploration. In the context of our sign language recognition project, Pandas is primarily used to manage and preprocess the dataset, making it easier to manipulate the data before feeding it into machine learning models.

Pandas is also used in the preprocessing phase to clean and prepare the dataset for training. This could involve normalizing or scaling the data, encoding categorical labels into numerical format (using functions like pd.get_dummies or LabelEncoder), and merging or joining data from different sources. It provides easy-to-use methods for handling missing data, filtering out irrelevant or corrupted entries, and converting the data into a suitable format (e.g., converting the images into NumPy arrays that can be fed into a deep learning model).

Moreover, after model training, Pandas is useful for evaluating the model's performance by analyzing the results. It allows easy calculation of evaluation metrics (such as accuracy, precision, and recall) and helps organize the results in a structured way to compare the performance of different models. Pandas can also be used for visualizing the training progress, such as plotting loss and accuracy curves during model training and validation.

vi. OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful, open-source library that provides tools and algorithms for real-time computer vision and image processing tasks. It is widely used in applications involving image and video manipulation, facial recognition, object detection, and more. OpenCV is highly efficient and optimized for both real-time applications and large-scale processing, making it a valuable tool in machine learning and computer vision projects, such as our sign language recognition system.

In this project, OpenCV plays a crucial role in collecting and processing the image data. Since the project involves recognizing hand gestures from real-time video input, OpenCV is used to capture video frames

from the webcam. Each frame is processed to detect and recognize the gestures corresponding to different letters of the American Sign Language alphabet. OpenCV helps in tasks such as resizing the captured frames, converting the images into the appropriate color format (e.g., from BGR to RGB), and performing operations like cropping or rotating images for alignment.

During real-time prediction, OpenCV is used to display the webcam feed and show the recognized letters or words on the graphical user interface (GUI). It can also be employed to track the movement of the hands and continuously process the video feed to predict the gesture in each frame. OpenCV integrates seamlessly with deep learning models, allowing for efficient image processing,

which is essential for achieving smooth and fast real-time sign language recognition.

vii. H5py

H5py is a Python library that provides an interface to interact with the HDF5 (Hierarchical Data Format version 5) file format. HDF5 is a popular file format for storing large amounts of data, particularly in scientific computing, and is widely used in machine learning and deep learning projects for storing model weights, training data, and other large datasets. The H5py library makes it easy to read and write HDF5 files, allowing users to efficiently manage and access large amounts of data without consuming excessive memory.

In the context of our sign language recognition project, H5py is primarily used for saving and loading trained model weights. When training deep learning models, such as the custom CNN, VGG16, or InceptionV3 models, the trained weights are typically stored in HDF5 files (with the .h5 extension). These files contain the weights and configuration of the neural network, which are essential for the model to make predictions. After training a model, H5py is used to save the model weights to an HDF5 file, and later, this file can be loaded to make predictions or resume training from where it was left off.

viii. Scikit-Learn

Scikit-learn (often referred to as sklearn) is a powerful and widely-used Python library for machine learning that provides a range of simple and efficient tools for data analysis and modeling. It offers implementations of a wide variety of machine learning algorithms for classification, regression, clustering, and dimensionality reduction, along with utilities for model evaluation and preprocessing. scikit-learn is primarily used for several key tasks, including data preprocessing, model evaluation, and splitting the dataset. For example, scikit-learn's **train_test_split** function is used to split the dataset into training and testing sets, ensuring that the model is trained on one portion of the data and evaluated on another. This is critical for avoiding overfitting and assessing how well the model generalizes to new, unseen data.

Additionally, scikit-learn provides tools for encoding the categorical labels in the dataset into numerical values using methods like **LabelEncoder** or **OneHotEncoder**. These tools are useful for preparing the data to be fed into deep learning models, which typically require numerical input and

output.Once the models (like custom CNN, VGG16, or InceptionV3) are trained, scikit-learn comes in handy for model evaluation. It provides a wide range of performance metrics such as **accuracy**, **precision**, **recall**, **F1-score**, and **confusion matrix** to evaluate the model's ability to classify hand gestures correctly. These metrics are crucial for comparing the performance of different models and fine-tuning their hyperparameters for better performance.

ix. CustomTkinter

CustomTkinter is an extension of the popular Tkinter library in Python, which provides a simple way to create graphical user interfaces (GUIs). While Tkinter offers basic UI components such as buttons, labels, and text fields, CustomTkinter enhances these elements with more modern and visually appealing designs, allowing for the creation of sleek, customizable interfaces with minimal effort. It is built on top of Tkinter but offers additional styling options, including the ability to adjust colors, fonts, and widget shapes, making it a good choice for creating polished applications.

In the context of our sign language recognition project, CustomTkinter is used to build the graphical user interface (GUI) for real-time sign language recognition. The GUI allows users to interact with the system, providing an intuitive interface to capture webcam input, display predictions, and visualize recognized gestures as text. CustomTkinter makes it easy to create a visually appealing layout with buttons, labels, and text boxes for the user to interact with, and it supports dynamic updates as predictions are made.

The flexibility of CustomTkinter allows us to integrate features such as displaying the webcam feed, showing the recognized hand gestures, and updating the displayed text in real-time as users perform sign language gestures. CustomTkinter also enables the design of interactive elements like buttons for starting the recognition process or resetting the system, making the application easy to use for both novice and experienced users.

Additionally, CustomTkinter's ability to customize the appearance of the interface, such as creating rounded corners, gradient backgrounds, and modern button designs, helps ensure that the GUI is user-friendly and professional-looking, enhancing the overall user experience.

3.4 Hardware Requirements

For a project involving deep learning models, real-time video processing, and GUI development, the hardware requirements need to support efficient data handling, model training, and interactive user interfaces. Below are the detailed hardware requirements necessary to develop and run the sign language recognition system effectively:

The **CPU (Central Processing Unit)** is a critical component for running the core tasks of the system, including handling software operations, data management, and general processing tasks. For the sign language recognition system, the CPU handles data preprocessing, managing video frame extraction, and running general software algorithms. A high-performance CPU, especially one with multiple cores, is essential for efficiently managing real-time data input from the webcam, ensuring that the system can process each video frame in a timely manner. While the CPU is not as crucial for deep learning model training (which benefits from the parallelism of a GPU), it plays a significant role in ensuring that the system runs smoothly without delays, especially during real-time inference tasks.

The **GPU (Graphics Processing Unit)** is required for handling the heavy computations involved in deep learning, particularly for training and inference tasks. Unlike the CPU, which is optimized for single-threaded operations, the GPU is designed to process large batches of data in parallel, making it indispensable for tasks like training complex models and running real-time predictions. For the sign language recognition project, the GPU is used to accelerate the training of models such as CNN, VGG16, and InceptionV3, which involve large-scale matrix operations. A dedicated GPU, such as an Nvidia GeForce GTX 1060 or higher, is recommended for significantly speeding up model training and inference times.

The **webcam** is an essential input device for capturing real-time video streams, which form the basis for gesture recognition in the system. For accurate sign language recognition, a high-quality webcam with at least 720p resolution and a frame rate of 30 fps is required to capture clear and smooth video of the user's hand gestures. The webcam allows the system to track and interpret the movement of hands, enabling real-time predictions. A better webcam with 1080p resolution or higher is preferred, as it provides sharper image quality and improves the system's ability to detect finer details in hand gestures. The webcam's performance is critical for ensuring that the captured frames are clear and accurate, which is key to achieving high accuracy in the model's predictions.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Evaluation of Trained Models

4.1.1 CNN

Accuracy : 99.94%

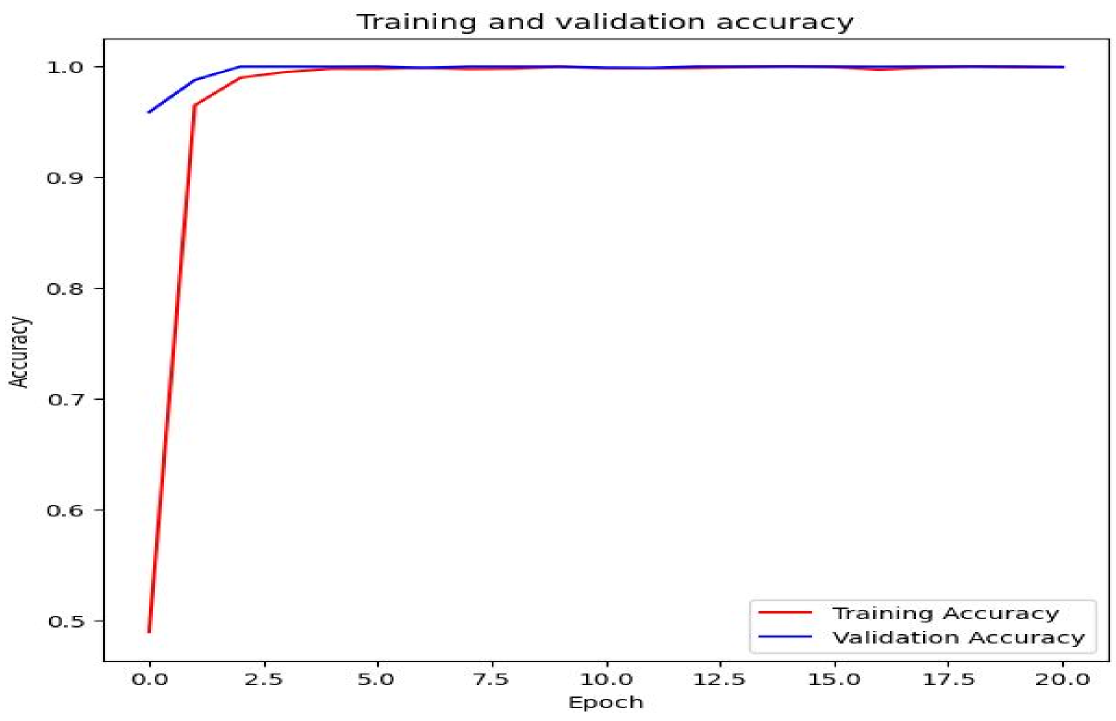


Fig 4.1 : Training and Validation Accuracy Graph (CNN)

The CNN model underwent training for 50 epochs, and early stopped at 20 epochs to prevent overfitting, with the following results achieved during the process. In the early stages, the model demonstrated rapid improvement, with the accuracy increasing from 21.40% in the first epoch to 94.90% in the second epoch. As training progressed, the model's accuracy continued to improve steadily, reaching an outstanding 99.94% by epoch 20.

The graph shows Training Accuracy and Validation Accuracy over 20 epochs for CNN model. The

model is performing very well with almost perfect accuracy on both training and validation data .There is no sign of overfitting as the validation accuracy follows closely with the training accuracy.

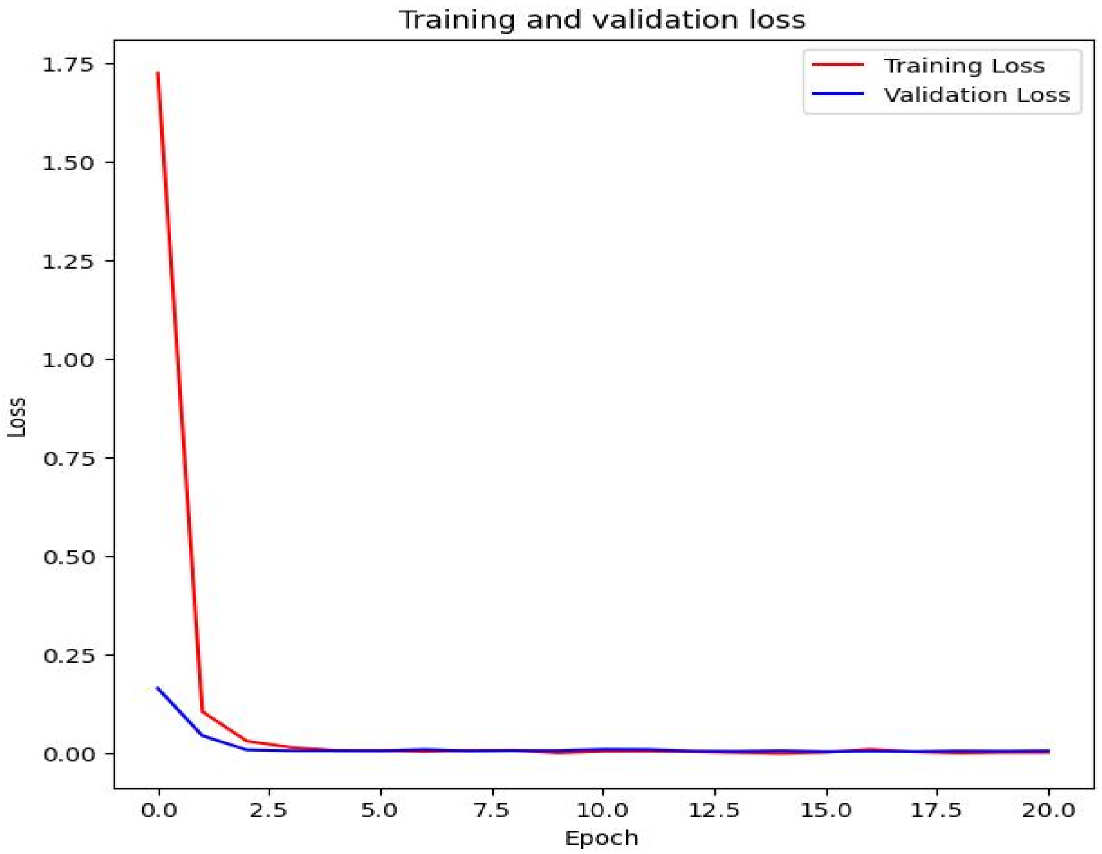


Fig 4.2: Training and Validation Loss Graph (CNN)

The training loss showed a corresponding decline, from 2.7178 to 0.0010 by epoch 21, indicating that the model was effectively learning and minimizing error over time.

In terms of validation performance, the model also showed excellent results. Initially, the validation accuracy was 95.85%, which then steadily improved, achieving 99.98% accuracy by epoch 4 and maintaining a high level of accuracy throughout the training. The validation loss exhibited a similar trend, decreasing from 0.1655 to 0.0053 by epoch 18, which suggests that the model was

generalizing well to unseen data.These results demonstrate that the CNN model successfully learned to classify the sign language gestures with high accuracy. The early improvement in accuracy and loss suggests that the model was able to quickly adapt to the dataset, while the sustained high performance indicates that the model is not overfitting and can generalize well to new, unseen data. The slight fluctuation in validation accuracy after epoch 19 may be attributed to factors such as minor variations in the data or model updates, but overall, the model performed very well in terms of both accuracy and loss reduction.

This performance, with such a high validation accuracy and low validation loss, confirms that the CNN model is well-suited for sign language recognition tasks, showing robust learning and generalization capabilities.

4.1.2 VGG16

Accuracy : 99.97%

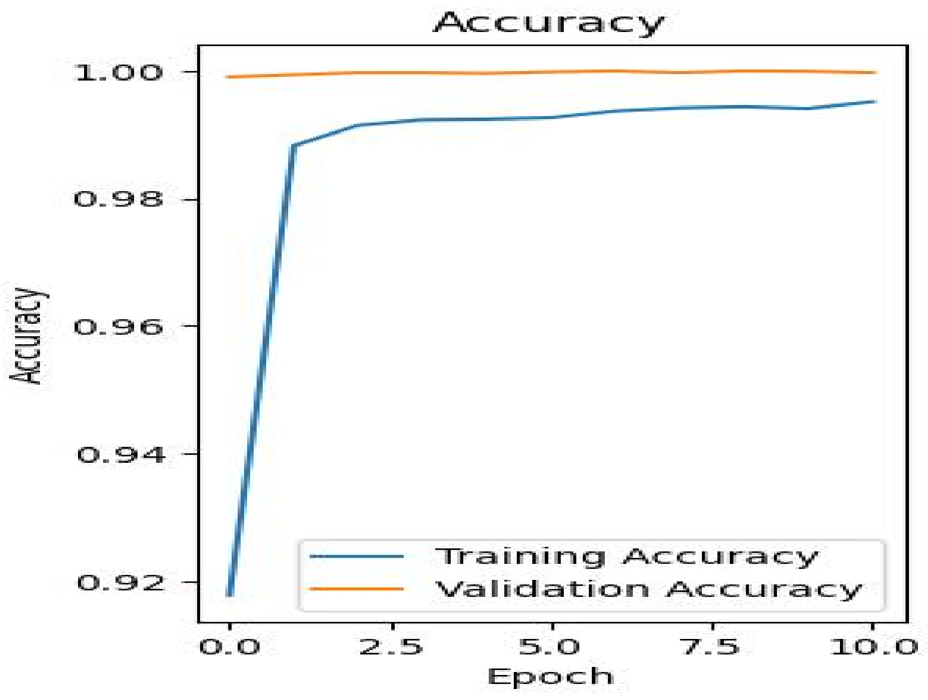
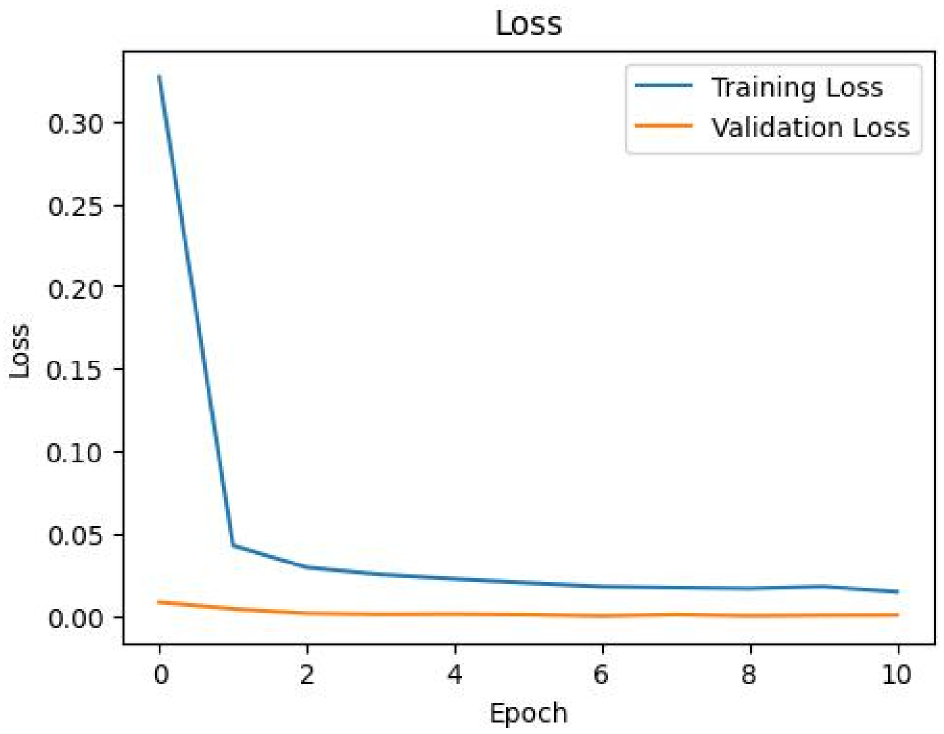


Fig 4.3 : Training and Validation Accuracy Graph (VGG16)

The graph shows **Training Accuracy** and **Validation Accuracy** over **10 epochs** for **VGG16**

model.The high validation accuracy from the start indicates that the model is performing exceptionally well on unseen data as well. No sign of **overfitting**, as the training and validation accuracy are very close to each .The accuracy of the VGG16 model demonstrates a high level of performance, with the training accuracy steadily improving from 91.78% in the first epoch to 99.97% by the 11th epoch. The model displayed strong learning capabilities, with the **validation accuracy** reaching 99.90% in the second epoch and maintaining a high level, peaking at **99.97%** by the 11th epoch. This indicates that the model was able to effectively learn the features of the data and generalize well to the validation set. After achieving 99% accuracy at epoch 11, the model maintained this perfect validation accuracy, suggesting that further training was unnecessary, and stopping early helped to prevent overfitting.



.

Fig 4.4 : Training and Validation Loss Graph (VGG16)

The training loss starts relatively high in the initial epoch, indicating a higher error in the model's predictions early in training. However, it rapidly decreases over the first few epochs, showing that the model is learning effectively. By around epoch 4, the training loss plateaus at a very low value, indicating that the model has achieved minimal error on the training set.

The validation loss starts low from the beginning and continues to decrease slightly, but it remains relatively stable after the first few epochs. This stability in validation loss, combined with its low values, suggests that the model is not overfitting, as there is no divergence between training and validation loss.

Overall, the low and consistent validation loss, coupled with the decreasing training loss, indicates that

the model has learned effectively and generalized well on the validation data. This balance reflects a well-optimized model with minimal risk of overfitting, confirming that training was stopped at an optimal point.

4.1.3 InceptionV3

Accuracy-94.24%

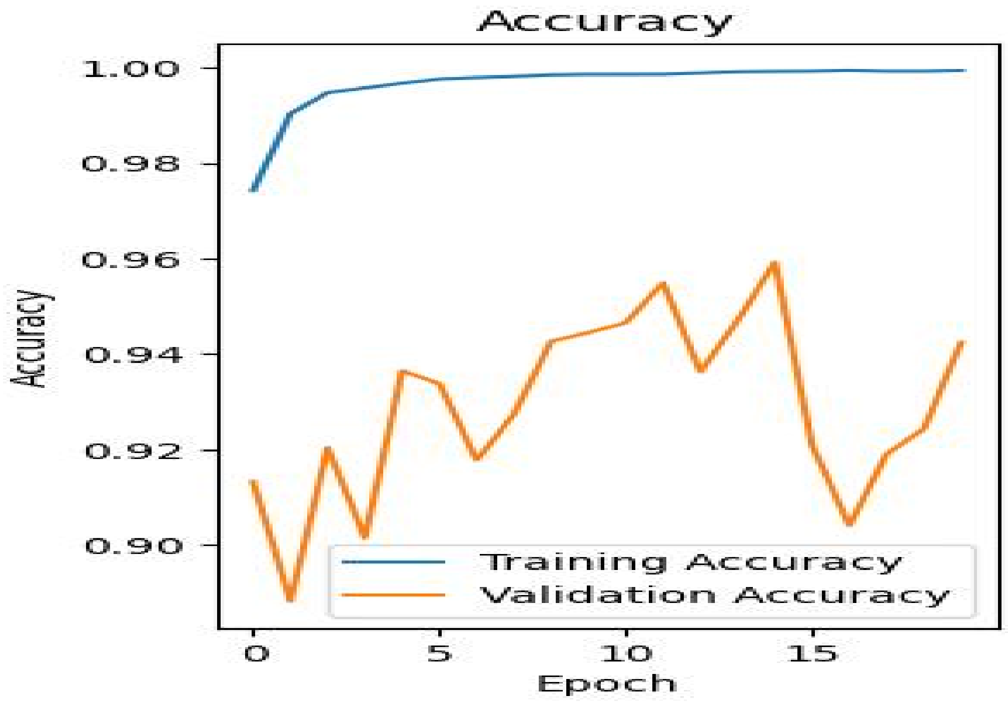


Fig 4.5 : Training and Validation Accuracy Graph (InceptionV3)

The training accuracy quickly rises and stabilizes near 100% within the first few epochs. **Early stopping** appears to be beneficial here, as it prevents the model from continuing to train in a state

where it's likely overfitting.In the first few epochs, the model's training accuracy is high, starting at 97.42% and improving steadily, while the validation accuracy fluctuates. This is typical as the model begins to learn patterns in the training data but generalizes variably on validation data.

The training accuracy consistently improves and reaches over 99%, indicating that the model is effectively learning the training data. The validation accuracy also improves, reaching a peak of 95.49% at epoch 12, showing the model's improved generalization ability to unseen data.

Validation accuracy fluctuates, reaching a high of 95.92% at epoch 15, but then decreases in subsequent epochs. Meanwhile, the training accuracy remains very high, indicating some overfitting. This gap suggests that while the model performs well on training data, it is less consistent on validation data.

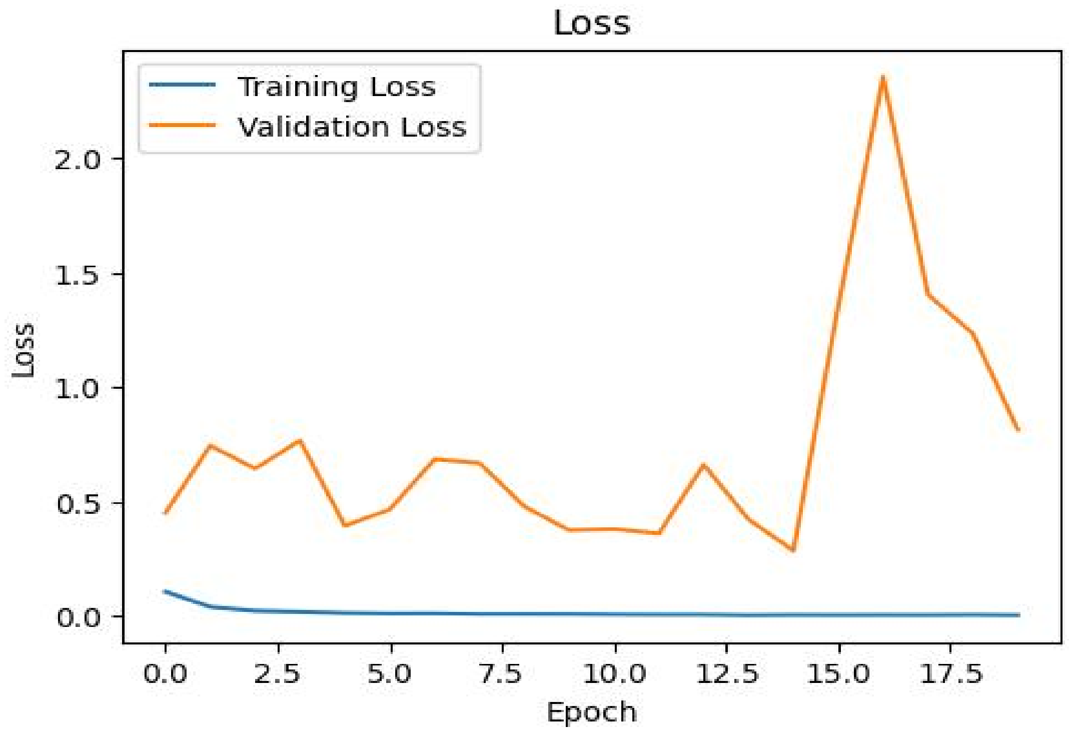


Fig 4.6: Training and Validation Loss Graph (InceptionV3)

The training loss starts relatively high at the beginning of training and decreases steadily over time, reaching very low values close to zero. This pattern shows that the model is learning the training data effectively and minimizing the error on this dataset.

Initially, the validation loss is higher than the training loss, as expected, but fluctuates in the early epochs. This variability suggests that the model is adjusting and trying to generalize on the validation set. Around the middle epochs, the validation loss shows a decline, indicating that the model is beginning to generalize better to the validation data.

However, in the later epochs (especially after epoch 15), there's a significant spike in validation loss, reaching values over 2. This sharp increase suggests **overfitting**, where the model is no longer generalizing well and is instead memorizing the training data patterns.

The **InceptionV3** model shows effective learning on the training set, evidenced by high training accuracy and low training loss. However, the increasing gap between training and validation performance in later epochs points to overfitting.

4.2 Comparative Analysis of Trained Models

The Custom CNN model achieved an impressive accuracy of **99.94%** on the dataset, which indicates that it performed well in learning the patterns within the training data. However, despite its high accuracy, Custom CNN might lack the complexity and depth of layers found in more sophisticated models like VGG16 and InceptionV3. This model's performance suggests that it can capture essential features for classification, but there may be limitations in its ability to generalize well to more complex or varied data, which typically requires deeper networks.

VGG16 demonstrated exceptional accuracy, reaching **99.97%**, which is slightly higher than the Custom CNN. This small but significant improvement indicates that VGG16's deeper architecture, consisting of multiple convolutional layers, allows it to extract more detailed and hierarchical features from the data. The stability of VGG16 during training (as observed from consistent accuracy and loss curves) suggests that it effectively balances learning complex patterns without overfitting. Given its superior accuracy and stability, VGG16 emerges as the most reliable model for this task, capable of high precision in classification with minimal error.

InceptionV3 reached an accuracy of **94.24%**, which is considerably lower than both Custom CNN and VGG16. This model, known for its complex architecture and multi-scale feature extraction through Inception modules, struggled with this dataset compared to the other models. The fluctuations in validation accuracy and spikes in validation loss imply that InceptionV3 faced

challenges in generalizing well, possibly due to overfitting or excessive model complexity relative to the dataset's needs. While InceptionV3 is generally effective in handling complex data, it may be less suitable for this specific task where simpler architectures.

Model	Training Accuracy	Validation Accuracy
CNN	**99.92%**	**99.94%**
VGG16	**99.51%**	**99.97%**
InceptionV3	**99.94%**	**94.24%**

Table 4.1 : Comparison of Trained Models

The results indicate that both the CNN and VGG16 models achieved exceptionally high accuracy, with VGG16 achieving the highest accuracy at 99.97% and CNN reaching 99.94%.InceptionV3, while performing slightly lower at 94.24%, still demonstrated reliable accuracy, showcasing the strength of these architectures in handling complex gesture recognition tasks.

Based on the analysis, **VGG16 is the best model** for this task due to its highest accuracy of 99.97% and consistent performance during training and validation. Its architecture is well-suited for the dataset, allowing it to learn effectively without overfitting. While Custom CNN also performed admirably with 99.94% accuracy, VGG16’s slightly higher accuracy and deeper feature extraction capabilities make it the top choice.

After thorough evaluation of the model performances, VGG16 has been identified as the most suitable choice for deployment in the desktop interface for real-time sign language recognition. With an impressive accuracy of 99.97%.

This makes VGG16 the optimal choice for the frontend, ensuring that the desktop interface provides users with accurate and responsive predictions. Therefore, VGG16 is selected for deployment, promising a robust and effective real-time prediction experience.

4.3 Screenshots

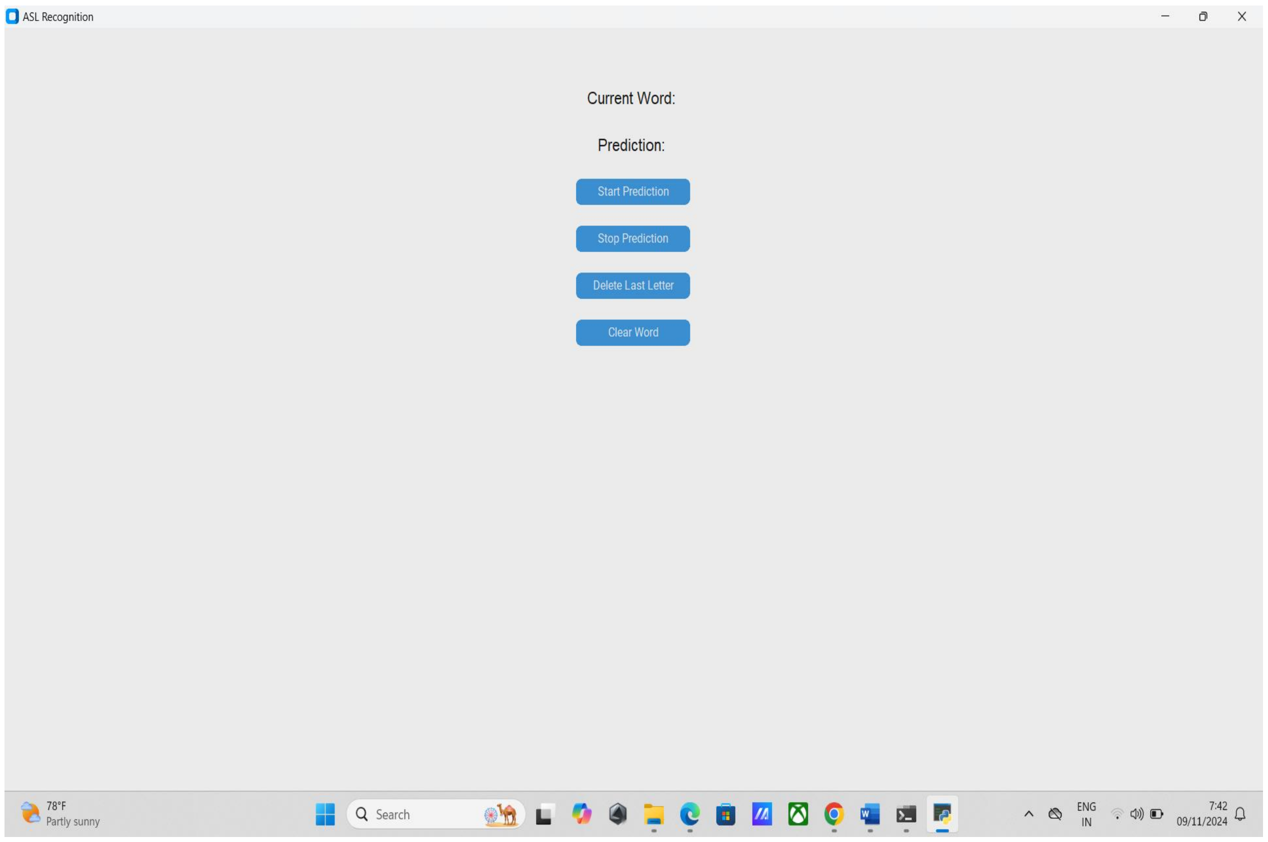
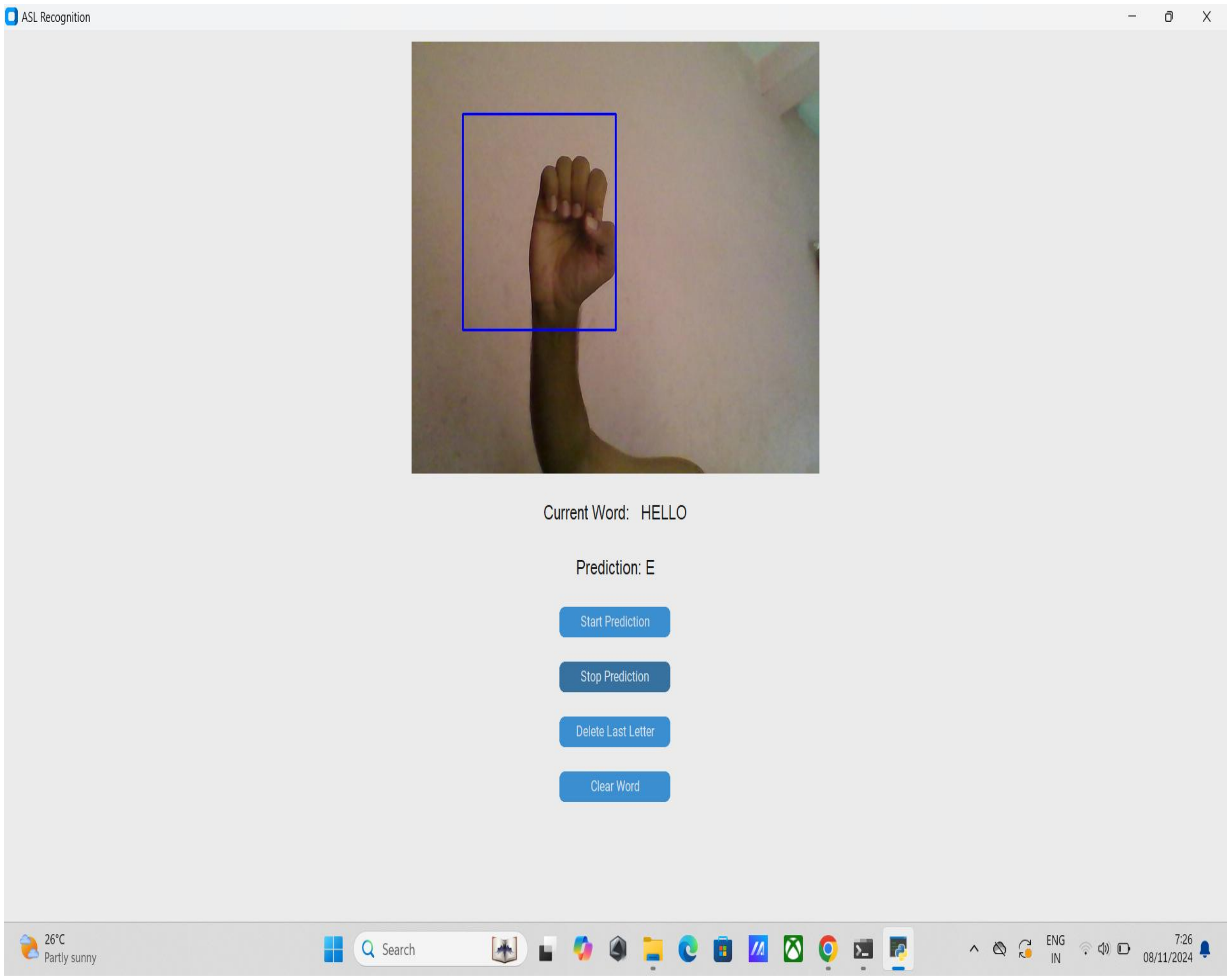


Fig 4.7 : Screenshot of Home Page



Current Word: HELLO

Prediction: E

Fig 4.8 : Screenshot of prediction page

CHAPTER 5

CONCLUSION

In this project, a comprehensive system for real-time American Sign Language (ASL) recognition has been developed, utilizing deep learning and pretrained models to achieve high accuracy and reliability. Three models—Custom CNN, VGG16, and InceptionV3—were trained and evaluated on a custom-collected dataset containing thousands of ASL alphabet samples. Among these models, VGG16 demonstrated the best performance with an accuracy of 99.97%, followed closely by the Custom CNN. The superior feature extraction capability of VGG16 proved to be crucial for accurately distinguishing between intricate ASL gestures, making it the most effective choice for this application.

The selected model, VGG16, has been integrated into a graphical user interface (GUI) designed for real-time prediction on a desktop platform. This GUI interface is crafted to be user-friendly and responsive, allowing users to input live video or images and receive immediate predictions on the displayed ASL letters. This real-time capability enables seamless interaction, providing a practical and efficient tool for ASL interpretation. The interface displays recognized gestures continuously, allowing the system to translate gestures into text in real-time, effectively bridging the communication gap for ASL users.

This project underscores the potential of deep learning in enhancing accessibility and inclusivity. By deploying the VGG16 model within a dedicated GUI for real-time ASL recognition, the system delivers both high precision and responsiveness.

5.1 FUTURE ENHANCEMENT

In future enhancements, the addition of a **speech output feature** would elevate the system's usability by providing spoken feedback of recognized ASL gestures. This feature would enable real-time conversion of sign language into audible speech, creating a seamless communication bridge for those interacting with ASL users. By implementing text-to-speech functionality, the system could verbally announce recognized letters or words, facilitating clearer communication for users who are unfamiliar with ASL or for visually impaired individuals who rely on auditory cues..Expanding the system to offer multilingual support is another valuable enhancement. With this feature, recognized ASL gestures could be translated into multiple languages, providing flexibility and accessibility for users from diverse linguistic backgrounds. This multilingual capability would make the system adaptable across different regions, offering support to ASL users who communicate with speakers of various languages. By implementing language translation modules, the system could translate ASL into other spoken or written languages, broadening its scope as a global assistive tool.

REFERENCES

[1] **KASAPBAŞI, Ahmed & Elbushra, Ahmed & AL-HARDANEE, Omar & YILMAZ, Arif**. (2022),DeepASLR: A CNN based Human Computer Interface for American Sign Language Recognition for Hearing-Impaired Individuals,Computer Methods and Programs in Biomedicine Update. 2. 100048. 10.1016/j.cmpbup.2021.100048.

[2] **Sundar, B & Thirumurthy, Bagyammal**. (2022),American Sign Language Recognition for Alphabets Using MediaPipe and LSTM. Procedia Computer Science. 215. 642-651. 10.1016/j.procs.2022.12.066.

[3] **Prof. Mrs. Maheshwari Chitampalli** el.at (2023). REAL TIME SIGN LANGUAGE DETECTION,International Research Journal of Modernization in Engineering Technology and Science.

[4] **Orovwode, Hope & Oduntan, Ibukun & Abubakar, John**. (2023), Development of a Sign Language Recognition System Using Machine Learning 1-8 10.1109/icABCD59051.2023.10220456.

[5] **Alsharif, Bader & Alalwany, Easa & Ilyas, Mohammad.** (2024). Transfer learning with YOLOV8 for real-time recognition system of American Sign Language Alphabet. Franklin Open. 8. 31. 10.1016/j.fraope.2024.100165.

[6] **Obi, Yulius & Claudio, Kent & Budiman, Vetri & Achmad, Said & Kurniawan, Aditya.** (2023), Sign language recognition system for communicating to people with disabilities. Procedia Computer Science. 216. 13-20. 10.1016/j.procs.2022.12.106.

.