PROJECT REPORT

on

AMERICAN SIGN LANGUAGE ALPHABET RECOGNITION

Submitted by

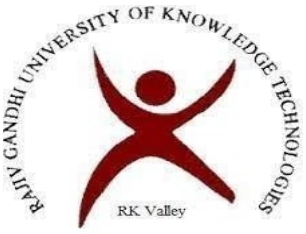
B. PRIYANKA
A. THULASI

Under the Guidance of

Dr. MUMMELA MUNI BABU

M.TECH, (Ph.D.), Assistant Professor

Department of Computer Science and Engineering



Rajiv Gandhi University of Knowledge and Technologies (RGUKT),
R.K.Valley, Kadapa, Andhra Pradesh, 516 330

DECLARATION

We BUKKE PRIYANKA and AKEPATI THULASI bearing ID:R201018 and R200574 hereby declares that the project report entitled “American sign language alphabet recognition” done under the guidance of Dr. MUMMELLA MUNI BABU sir is submitted in partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering during the academic year 2024 – 2025 at RGUKT-RK Valley. I also declare that this project is a result of our efforts. Citations from any websites are mentioned in the references.

Date: B. Priyanka (R201018)

Place: RK Valley A. Thulasi (R200574)

CERTIFICATE

This is to certify that the project work entitled "**American sign language alphabet recognition**" is a bonafide project work submitted by **B.PRIYANKA(R201018) , A.THULASI(R200574)** in the Department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering for the year 2024- 2025 carried out the work under our guidance and supervision.

Project Guide

Dr. MUMMELA MUNI BABU
Assistant Professor
Computer Science and Engineering
RGUKT, RK Valley

Head of the Department

Dr. RATNAKUMARI CHALLA
Head of the Department
Assistant Professor
RGUKT, RK Valley

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude & respect to all those people behind the screen who guided, inspired, and helped us crown all our efforts with success. I wish to express my gratitude to **Dr. MUMMELA MUNI BABU** for his valuable guidance at all stages of study, advice, constructive suggestions, supportive attitude, and continuous encouragement, without which it would not be possible to complete this project.

I would also like to extend my deepest gratitude & reverence to the Director of RGUKT, RK Valley **Dr. A V S Kumara Swami Gupta,** and HOD of Computer Science and Engineering **Dr. Ch. Ratna Kumari** for their constant support and encouragement.

Last but not least I express my gratitude to my parents for their constant source of encouragement and inspiration for me to keep my morals high.

With Sincere Regards,

BUKKE PRIYANKA(R201018)
AKEPATI THULASI(R200574)

Table of Contents

ABSTRACT

Human-Computer Interaction (HCI) is making great progress in gesture-based communication, especially in American Sign Language (ASL) recognition. Sign language helps individuals with hearing impairments communicate with others, and computers can assist in this process. Compared to other forms of gesture-based communication, ASL recognition has gained more attention from researchers due to its practical applications.

This study focuses on a vision-based system that recognizes hand gestures, as hand movements are key to sign language. Traditional methods using special hardware have some challenges, but computer vision and pattern recognition offer better solutions. The main goal of this research is to identify hand movements accurately using Convolutional Neural Networks (CNNs).

CNNs are powerful tools for image recognition. They can analyze pictures by detecting important features like edges and patterns, making them ideal for recognizing hand gestures in ASL. Originally developed for image processing, CNNs can also be used for analyzing sequences of movements, improving the accuracy of sign language detection.
By using Artificial Intelligence (AI) and deep learning, this study contributes to the field of Human-Computer Interaction (HCI), helping to bridge the communication gap for people with hearing impairments through accurate and efficient ASL recognition.

Keywords: Convolutional Neural Networks, Deep Learning, Image Processing, American Sign Language Recognition, Human-Computer Interaction.

LIST OF FIGURES

LIST OF TABLES

CHAPTER -1

INTRODUCTION:

Humans can communicate with one another in a variety of ways. This include behaviour. such as physical gestures, facial expressions, spoken words, etc. However, those who have hearing loss are restricted to using handgestures to communicate. /or speech impairments communicate using a standard sign language that is incomprehensible to non-users.

Sign language is the communication system for those who are hard of hearing and deaf. It ranks as the sixth most utilized language worldwide. It is a type of communication that uses hand movements to communicate ideas. Eachregion has its specific sign language like normal language. In 2005 there were an estimated 62 million deaf people worldwide and about 200 different sign languages in use around the world, many of which have distinctive features.

ASL is the primary language of many deaf citizens in North America. Hard-of-hearing and hearing people also use it. Hand gestures and facial expressions are used to convey this language. The deaf community has access to ASL as a means of communication with the outside world and inside the community. But not everyone is familiar with the signs and motions used in sign language.

Understanding sign language and being familiar with its motions takes a lot of practice. Since there are no reliable, portable tools for identifying sign language, learning sign language takes a lot of time. However, since the development of neural networks and deep learning, it is now possible to create a system that can identify things, or even objects of different catogories.

Problem Statement

It's important to interact with everyone in our modern culture, whether it's for fun or for work. Communication has always had a great impact in every domain and how it is considered the meaning of the thoughts and expressions that attract the researchers to bridge this gap for every living being.

Speech impairment is a disability which affects an individuals ability to communicate using speech and hearing. People who are affected by this use other media of communication such as sign language. However, learning and understanding sign language requires a lot of practice, and not everyone will comprehend what the gestures in sign language indicate. It takes time to learn sign language because there is no reliable, portable tool for doing so.

Hearing or speech-impaired individuals who are proficient in sign language need a translator who is equally proficient in sign language to 4 effectively communicate their ideas to others. This technique assists people with hearing loss or speech impairments in learning and translating their sign language in order to help them overcome these issues.

CHAPTER-2

LITERATURE REVIEW:

Table1: Literature Review

Ref	Author	Year	Methodology	Dataset Used	Findings	Disadvantage
1.	Kaggle ASL Dataset Research	2023	ASL Hand Gesture Recognition	ASL Alphabet Dataset (87,000 images)	Achieved 99% accuracy in classifying 26 letters of ASL	Convolutional Neural Network (CNN) Limited to static images, lacks dynamic gesture recognition
2.	S. Pigou et al.	2016	Isolated Sign Gesture Recognition	RWTH-1) PHOENIX-2) Weather 3) dataset	CNN extracts spatial features, RNN captures temporal dependencies	Limited dataset, complex real-world scenario
3.	J. Huang et al.	2018	ASL Fingerspelling Recognition	American Sign Language Lexicon Video Dataset ASLLVD	Combined spatial-temporal modeling for improved accuracy	Requires large datasets and computational resources
4.	R. Rastgoo et al.	2020	Deep Learning for ASL Recognition	RWTH-BOSTON-400 Dataset	Transfer learning improved accuracy	Lacks real-time application validation

5.	R. Rastgoo et al.	2020	Deep Learning for ASL Recognition	RWTH-BOSTON-400 Dataset	Transfer learning improved accuracy	Lacks real-time application validation
6.	Wang et al.	2019	Real-time ASL Recognition	Custom dataset (hand gestures captured using webcams)	Achieved real-time hand gesture detection with 85% accuracy	Lower accuracy for complex gestures and low-light conditions
7.	A. Kumar et al.	2022	Static and Dynamic ASL Recognition	ASL Alphabet and Digits Dataset	Combined spatial and temporal features for better classification	Needs larger dataset for dynamic signs
8.	Zafer Ahmed Ansari	2020	Indian Sign Language	150 classes (fingerspelling numbers, common phrases, alphabet)	Identified hand using depth data, used unsupervised learning	Could not train using neural networks
9.	Divya Deora	2023	Sign Symbol Recognition	15 images per sign (small dataset)	segmentation and fingertip analysis led to 60% accuracy	Low accuracy due to poor camera quality and small dataset

Motivation

American Sign Language (ASL) is the primary mode of communication for millions of Deaf and hard-of-hearing individuals in the United States and Canada. As technology continues to evolve, there's a growing opportunity to bridge the gap between the hearing and Deaf communities through innovative solutions. The development of American Sign Language (ASL) recognition systems using artificial intelligence (AI) and machine learning is a promising avenue to create a more inclusive society.

In this context, the motivation behind pursuing an ASL recognition project is both personal and societal, driven by a desire to empower people with hearing impairments and foster better communication across different communities.

Bridging Communication Barriers

ASL recognition can provide a solution to one of the most significant barriers that Deaf individuals face in everyday life: communication. Although technology has come a long way in translating spoken language, the same cannot be said for ASL, which has a unique syntax and structure distinct from spoken languages. The inability to communicate seamlessly with the hearing population can result in feelings of isolation and frustration. By developing an accurate ASL recognition system, we can provide a tool that enables Deaf individuals to communicate with non-signers, enhancing social inclusion.

2. Promoting Equal Opportunities

- In education, healthcare, and other sectors, access to information and communication can be life-changing for Deaf individuals. However, the lack of ASL interpreters, particularly in remote or underfunded areas, often limits their ability to fully participate in various services. A robust ASL recognition system would empower Deaf individuals to access information independently, without relying on interpreters or being dependent on others. This autonomy can enhance their educational experience, improve access to healthcare, and help them integrate better into society, leveling the playing field for all.

3. Advancements in Artificial Intelligence (AI)

- This project also taps into the field of AI and machine learning, both of which

have seen tremendous growth in recent years. By applying computer vision, natural language processing, and deep learning techniques to recognize and translate ASL, the project provides an excellent opportunity to push the boundaries of AI. It combines cutting-edge technologies in real-time communication, image recognition, and data analysis, giving you hands-on experience with multiple aspects of AI, from data preprocessing to model deployment.

4. Real-Time Application and Practicality

- One of the most exciting aspects of this project is its potential for real-time application. Unlike traditional text-based translation systems, real-time ASL recognition allows for spontaneous, fluid communication. This can be a game-changer for public spaces, events, classrooms, or even casual conversations. The development of such systems opens up avenues for creating applications that could serve as live interpreters, facilitating daily interactions between hearing and non-hearing individuals.

5. Personal Growth and Societal Impact

- On a personal level, working on this project offers an opportunity to grow as a technologist. By combining your passion for technology with a meaningful purpose, you can contribute to something that has a direct, positive impact on society. This project provides a platform to learn and apply new skills in computer vision, deep learning, and web development, while also addressing real-world challenges. In doing so, you become part of a larger movement toward inclusivity and accessibility, making technology more equitable for all.

6. Potential for Further Innovation

- Beyond the immediate goal of ASL recognition, the techniques and frameworks developed in this project have applications in broader areas of human-computer interaction. ASL recognition can be a stepping stone toward developing gesture-based control systems for other domains, such as gaming, healthcare, and assistive technologies. By developing the ASL recognition system, you lay the groundwork for future innovations that can further enhance accessibility for individuals with disabilities.

Contribution

The contribution of the ASL alphabet recognition project using Convolutional Neural Networks (CNN) significantly advances accessibility, education, and communication support for the Deaf and hard-of-hearing community. By automating the recognition of hand gestures corresponding to ASL alphabets, the system offers a powerful tool for bridging communication gaps and promoting inclusivity.

1. Enhanced Accessibility for the Deaf Community:

The project contributes to improved accessibility by allowing real-time recognition of ASL alphabets through a camera-based system. This can help Deaf individuals communicate basic information through finger spelling, especially in environments where interpreters or text-based alternatives are unavailable.

2. Educational Support for ASL Learners:

The system can serve as an interactive tool for students and educators involved in learning or teaching ASL. It provides instant feedback on hand gesture recognition, which encourages independent practice and improves learning outcomes for beginners in sign language.

3. Efficient Use of CNN for Gesture Recognition:

The project employs a custom-trained Convolutional Neural Network optimized for classifying static hand signs representing the 26 letters of the English alphabet in ASL. The use of CNN's enables accurate feature extraction and classification, demonstrating the effectiveness of deep learning in visual gesture recognition tasks.

4. Real-Time Application Potential:

By ensuring low-latency prediction using lightweight CNN architecture, the system can be integrated into real-time applications such as ASL learning platforms, communication aids, and mobile apps. This improves the usability of the model in everyday contexts.

5. Automation of Manual Translation Tasks:

This project reduces the manual effort required to interpret ASL alphabets by automating gesture-to-text translation. This can be particularly beneficial in scenarios like

customer service, education, or healthcare, where time and clarity are essential.

6. Promoting Digital Inclusivity:

The system supports digital inclusion by making technology more accessible to a wider audience. It provides a stepping stone toward building more advanced sign language recognition tools that could eventually include full words, sentences, and facial expressions.

7. Open-Source and Research Contribution:

By sharing model architecture, training methodology, and dataset links, the project contributes to the research community, encouraging further development and innovation in sign language recognition using deep learning.

Background Work:

American Sign Language (ASL) is a complete visual language, where hand gestures represent words and letters. Recognizing ASL automatically using computer vision is important for developing assistive technologies for people with hearing and speech impairments.

Hand gesture recognition systems usually involve:

- Image acquisition (capturing hand signs)
- Preprocessing (resizing, normalizing)
- Feature extraction (using CNNs)
- Classification (predicting the correct letter)

Deep learning, especially Convolutional Neural Networks (CNNs), has significantly improved the accuracy of gesture recognition tasks by learning important spatial features directly from image data without manual feature engineering.

CHAPTER-3

CNN Deep Learning Model:

Authors:

Convolutional Neural Networks (CNNs) were first introduced by Yann LeCun in the late 1980s and popularized with the development of LeNet-5. CNNs are now a cornerstone of deep learning models in computer vision, with further advancements contributed by researchers worldwide through architectures like AlexNet, VGG, ResNet, and Inception.

Size and Parameters:

CNNs vary widely in size depending on their architecture. Simple CNNs for tasks like digit or alphabet recognition may have fewer than 1 million parameters, while deeper architectures like VGG or ResNet can have tens to hundreds of millions of parameters. The model size is determined by the number of layers, filters, and fully connected nodes used.

Overview:

CNN (Convolutional Neural Network) is a class of deep learning models specifically designed for processing structured grid data like images. Unlike traditional neural networks, CNNs use convolutional layers that apply learnable filters to local receptive fields of the input image, enabling the model to detect low- to high-level visual features such as edges, shapes, and textures.

CNNs are widely used in computer vision tasks such as image classification, object detection, face recognition, and medical imaging. Their ability to automatically extract and learn spatial hierarchies of features makes them highly effective for visual pattern recognition.

Merits:

1. **Automatic Feature Extraction:**

CNNs eliminate the need for manual feature engineering by learning spatial features directly from image data during training.

2. **High Accuracy in Vision Tasks:**

CNNs consistently achieve state-of-the-art results in image classification, segmentation, and detection challenges.

3. **Parameter Sharing and Sparsity:**

The use of shared weights in convolutional layers significantly reduces the number of parameters and computational cost compared to fully connected networks.

4. **Hierarchical Feature Learning:**

CNNs learn features in layers — starting from simple edges and corners to complex patterns like letters or objects.

5. **Scalability:**

CNN architectures are modular and scalable, allowing for customization depending on the complexity of the task and dataset.

Demerits:

1. **Computationally Intensive:**

Training deep CNNs requires high-performance GPUs and large memory, especially with high-resolution images and large datasets.

2. **Data-Hungry:**

CNNs typically require large amounts of labeled data for effective training and generalization.

3. **Lack of Contextual Understanding:**

CNNs focus on spatial features but lack the ability to understand sequential or contextual relationships unless combined with models like RNNs or Transformers.

4. **Vulnerability to Overfitting:**

Without proper regularization (e.g., dropout, augmentation), CNNs can overfit especially on small datasets.

CNN Architecture:

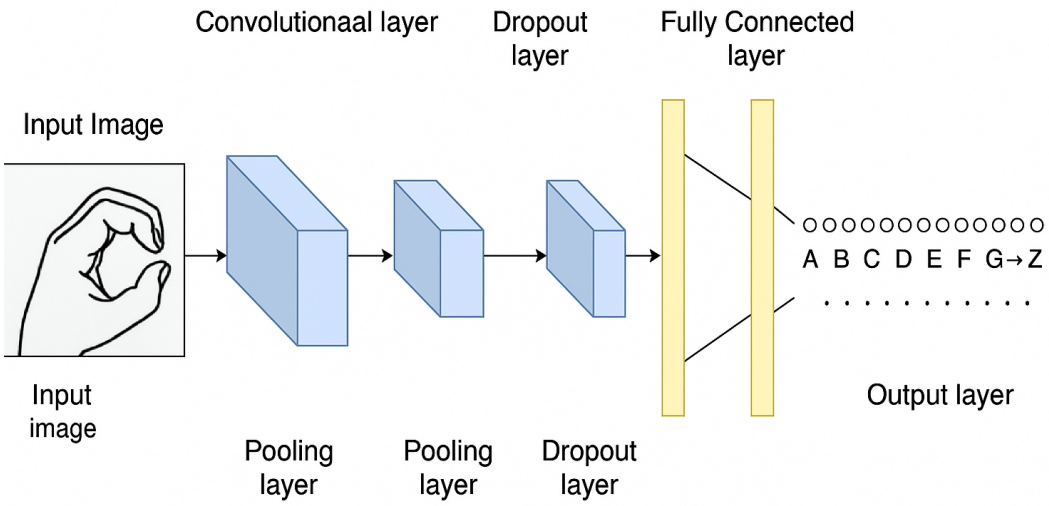


Figure 3.2 CNN Model Architecture for ASL Alphabet Recognition

The Convolutional Neural Network (CNN) used for ASL alphabet recognition is a deep learning model specifically designed to process image data of hand gestures. It extracts hierarchical visual features and classifies them into one of the 26 English alphabet classes. The architecture typically includes the following key components:

1. Input Layer:

Function: Accepts the input image (e.g., 64x64 or 128x128 pixel RGB or grayscale hand sign image).

Preprocessing: Normalizes pixel values (e.g., scales between 0 and 1) and resizes the image to a fixed dimension for model consistency.

2. Convolutional Layers:

Function: Apply learnable filters to the input image to detect spatial features such as edges, textures, and shapes.

Operation: Each convolution operation outputs a feature map that highlights specific patterns in the image.

Activation Function: Uses ReLU (Rectified Linear Unit) to introduce non-linearity.

3. Pooling Layers (Max Pooling):

Function: Downsample the feature maps, reducing dimensionality and computational cost while retaining the most important features.

Example: A 2x2 max pooling layer reduces a 32x32 feature map to 16x16.

4. Dropout Layers:

Function: Randomly deactivate a fraction of neurons during training to prevent overfitting and improve generalization.

5. Fully Connected (Dense) Layers:

Function: Flatten the extracted features from the convolutional layers and pass them through one or more dense layers to perform classification.

Operation: The final dense layer uses Softmax Activation to output a probability distribution over 26 classes (A–Z).

6. Output Layer:

Function: Produces the final classification result — the predicted ASL alphabet corresponding to the hand gesture.

Methodology

This project is divided into two main phases:

- **Phase 1: Model Development** (Training the model)
- **Phase 2: Model Deployment** (Building a real-time communication system)

Each phase consists of multiple clearly defined steps, explained below:

Phase 1: Model Development

1. Dataset Collection

The dataset consists of images of hand gestures representing the 26 alphabets (A-Z) ofAmerican Sign Language (ASL).

Each alphabet is stored in its respective folder. The dataset is divided into:

- Training Set: Used to train the model.
- Testing Set: Used to validate the model's performance on unseen data.

2. Data Preprocessing

To prepare the images for the CNN model:

- **Rescaling:**

  Every image pixel value is scaled between 0 and 1 by dividing by 255. This helps the neural network converge faster.

- **Data Augmentation:**

  Random transformations are applied on the training images to artificially expand the dataset and improve the model's generalization. The augmentations include:

  Shear transformations

  Zooming

  Horizontal flipping

- **Image Resizing**

  All images are resized to (64 × 64 pixels) to maintain a consistent input size for the CNN.

3. Model Building

A Convolutional Neural Network (CNN) was built using TensorFlow and Keras to automatically learn features from the ASL images.

Model Architecture:

- **Convolutional Layers:**

Three convolutional layers with 64, 128, and 256 filters, respectively, each using 3×3 kernels and ReLU activation.

- **Max Pooling Layers:**

After each convolutional block, MaxPooling2D layers are used to reduce the spatial dimensions, keeping important features.

- **Flatten Layer:**

Converts the 3D feature maps into a 1D feature vector to pass into dense layers.

- **Fully Connected Dense Layers:**
  - 512 neurons with ReLU activation.
  - Dropout of 0.5 applied to prevent overfitting.
- **Output Layer:**

26 neurons (one for each alphabet) with softmax activation to produce a probability distribution across all classes.

4. Model Training

The model is compiled and trained with the following configurations:

- **Optimizer:** Adam optimizer for efficient training.
- **Loss Function:** Categorical Crossentropy (suitable for multi-class classification).
- **Metrics:** Accuracy to monitor performance.

Training is done for up to 50 epochs but with **EarlyStopping** based on validation loss:

- If validation loss does not improve for 5 consecutive epochs, training stops to prevent overfitting.
- The best weights during training are automatically restored.

5. Model Evaluation

The model's performance is evaluated using:

- Training Accuracy and Loss Curves
- Validation Accuracy and Loss Curves

These curves help visualize whether the model is overfitting, underfitting, or performing well.

6. Model Saving

The best performing model is saved as:

- asl_better_model.h5 → Trained model file
- class_indices.json → Mapping between classes and label names

These saved files are used in the deployment phase.

Phase 2: Model Deployment (Real-Time Communication System)

1. Flask Web Application

A Flask web app is developed to provide an interface where users can:

- Upload an Image of a hand sign and predict the corresponding ASL alphabet.
- Use Webcam Live Feed to recognize ASL hand gestures in real-time.

Templates like home.html, upload.html, webcam.html, and result.html are used to create the user interface.

2. Image Upload Prediction

Users can upload an image:

- The image is resized to (64×64), normalized, and fed into the CNN model.
- The model predicts the alphabet with the highest probability.
- The prediction result is displayed along with the uploaded image.

3. Real-Time Webcam Prediction

Users can enable their webcam to perform live detection:

- A Region of Interest (ROI) box is displayed on the webcam feed.
- The ROI (hand area) is extracted, resized, normalized, and fed into the CNN model frame-by-frame.
- Predictions are shown live on the screen with:
    - Predicted Class

• Confidence Score (Only if confidence > 70%, otherwise a “waiting for clearer sign” message is shown)

4. Real-Time Communication

The final system allows:

• Continuous prediction through live video stream.

• Display of the recognized ASL alphabet in real time.•

• Accepting both webcam and uploaded images for prediction.

Figure 1.2 : Different Signs For Alphabets

CHAPTER-4

Module-1:Fine Tune Model

Step 1: Importing Neccessary packages and libraries.

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
```

Step 2 : Data Preprocessing and Augmentation using ImageDataGenerator

```
train_path = r"C:\Users\thula\Documents\Mini_Project\Dataset\training_set"
test_path = r"C:\Users\thula\Documents\Mini_Project\Dataset\test_set"

train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

train_set = train_datagen.flow_from_directory(train_path,
                                              target_size=(64,64),
                                              batch_size=32,
                                              class_mode='categorical')

test_set = test_datagen.flow_from_directory(test_path,
                                            target_size=(64,64),
                                            batch_size=32,
                                            class_mode='categorical')
```

```
Found 78000 images belonging to 26 classes.
Found 26 images belonging to 26 classes.
```

Step 3 : Save Class Indices(Label Mapping)

```
import json

with open("class_indices.json", "w") as f:
    json.dump(train_set.class_indices, f)

train_set.class_indices
```

Step 4 : Build The CNN Model

```
model = Sequential([
    Conv2D(64, (3,3), activation='relu', input_shape=(64,64,3)),
    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Conv2D(256, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(26, activation='softmax')  # 26 classes for A-Z
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Step 5 : Train the Model With Early Stopping

```
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau

# Define EarlyStopping
early_stopping = EarlyStopping(
    monitor='val_loss',     # what to monitor
    patience=5,             # how many epochs to wait before stopping
    restore_best_weights=True  # restore model weights from the epoch with the best value of the monitored quantity
)



# Fit model with EarlyStopping
history=model.fit(
    train_set,
    steps_per_epoch= 120,
    epochs=50,
    validation_data=test_set,
    callbacks=[early_stopping]  # Add the callback here
)
```

Step 6 : Save Model and Plot Accuracy

```
model.save('asl_better_model.h5')

plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc='lower right')
plt.show()
```

Step 7 : Evalate the Model On Test Set

```
# Evaluate the model on the test set
loss, accuracy = model.evaluate(test_set)

# Print accuracy as a percentage
print(f"Overall Test Accuracy: {accuracy * 100:.2f}%")
```

Module – 2: Integration Of Flask Web Application With Real-Time ASL Recognition Model

Step 1 : Import Required Libraries

```
from flask import Flask, render_template, Response, request
import cv2
import numpy as np
from keras.models import load_model
from keras.utils import img_to_array
import json
import os
```

Step 2 : Initialize Flask app and Load Model

```python
app = Flask(__name__)

# Load the trained ASL model
model = load_model("C:/Users/thula/Documents/Mini_Project/Training File/asl_bette

# Load class label mappings
with open('class_indices.json', 'r') as f:
    class_indices = json.load(f)
index_to_label = {v: k for k, v in class_indices.items()}

# Ensure uploads folder exists
os.makedirs('static/uploads', exist_ok=True)
```

Step 3 : Define Routes For Web Pages

```python
# Home Page
@app.route('/')
def index():
    return render_template('home.html')

# Webcam Prediction Page
@app.route('/webcam')
def webcam():
    return render_template('webcam.html')

# Image Upload Page
@app.route('/upload')
def upload():
    return render_template('upload.html')
```

Step 4 : Image Upload Prediction Logic

```python
# Image Prediction Route
@app.route('/predict_image', methods=['POST'])
def predict_image():
    if 'file' not in request.files:
        return "No file uploaded."
    file = request.files['file']
    if file.filename == '':
        return "No selected file."

    if file:
        filepath = os.path.join('static/uploads', file.filename)
        file.save(filepath)

        img = cv2.imread(filepath)
        img = cv2.resize(img, (64, 64))
        img = img / 255.0
        img = np.expand_dims(img, axis=0)

        prediction = model.predict(img, verbose=0)
        predicted_class_index = np.argmax(prediction, axis=1)[0]
        predicted_label = index_to_label[predicted_class_index]

        return render_template('result.html', label=predicted_label, image=filepath)
```

Step 5 : Real-Time Webcam Prediction Generator

```python
# Real-Time Frame Generator
def gen_frames():
    camera = cv2.VideoCapture(0)
    try:
        while True:
            success, frame = camera.read()
            if not success:
                break

            frame = cv2.flip(frame, 1)
            x1, y1, x2, y2 = 300, 100, 600, 400
            roi = frame[y1:y2, x1:x2]
            roi_resized = cv2.resize(roi, (64, 64))
            roi_rgb = cv2.cvtColor(roi_resized, cv2.COLOR_BGR2RGB)
            roi_array = img_to_array(roi_rgb) / 255.0
            roi_array = np.expand_dims(roi_array, axis=0)

            prediction = model.predict(roi_array, verbose=0)
            predicted_class = index_to_label[np.argmax(prediction)]
            confidence = np.max(prediction) * 100
            threshold = 70

            if confidence > threshold:
                display_text = f"Predicted: {predicted_class}"
                display_conf = f"Confidence: {confidence:.2f}%"
                color = (0, 0, 255)
            else:
                display_text = "Waiting for clearer sign..."
                display_conf = None
                color = (255, 165, 0)

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.putText(frame, display_text, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 1.3, color, 3)
            if display_conf:
                cv2.putText(frame, display_conf, (10, 80), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2)

            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()

            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
    finally:
        camera.release()
        print("Camera released")
```

Step 6 : Route to Stream Live Video to Browser

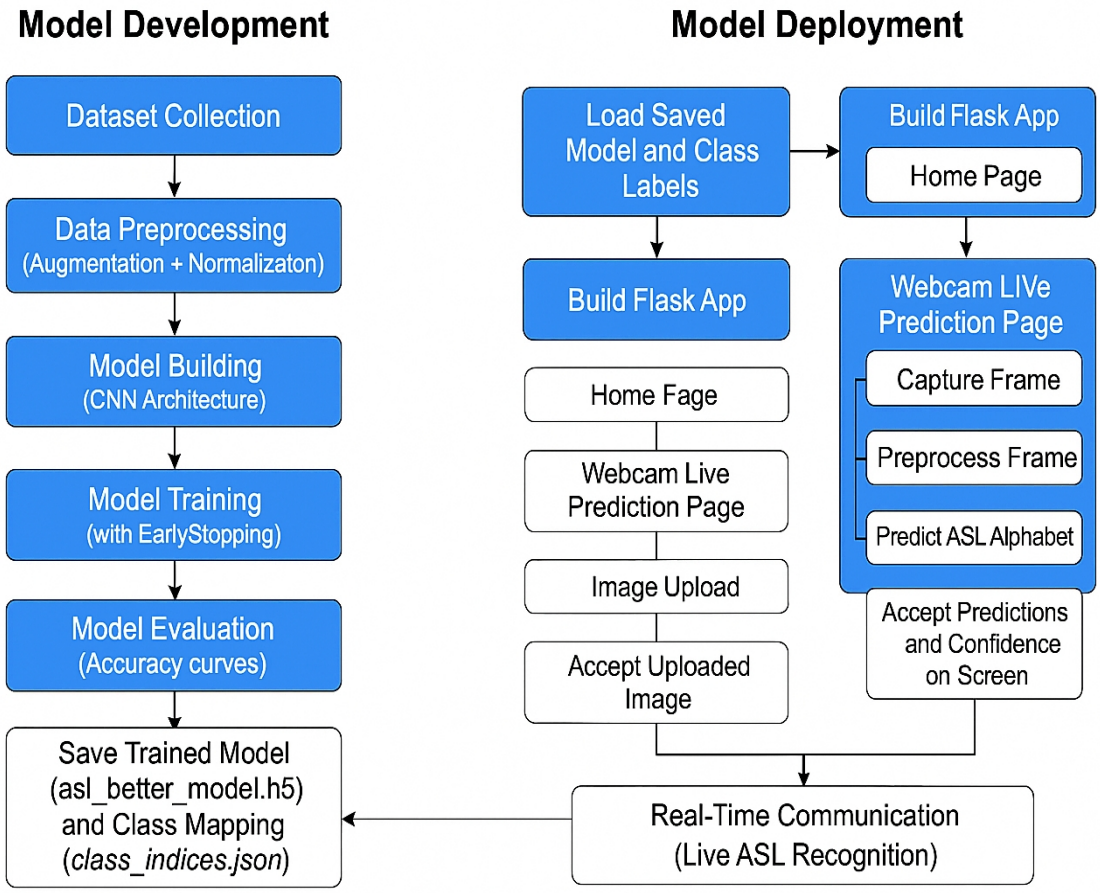
```python
# Video Streaming Route
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

Step 7 : Run the Flask App

```python
# Run the App
if __name__ == '__main__':
    app.run(debug=True)
```

CHAPTER-5

Flowchart of Project Implementation.



CHAPTER-6

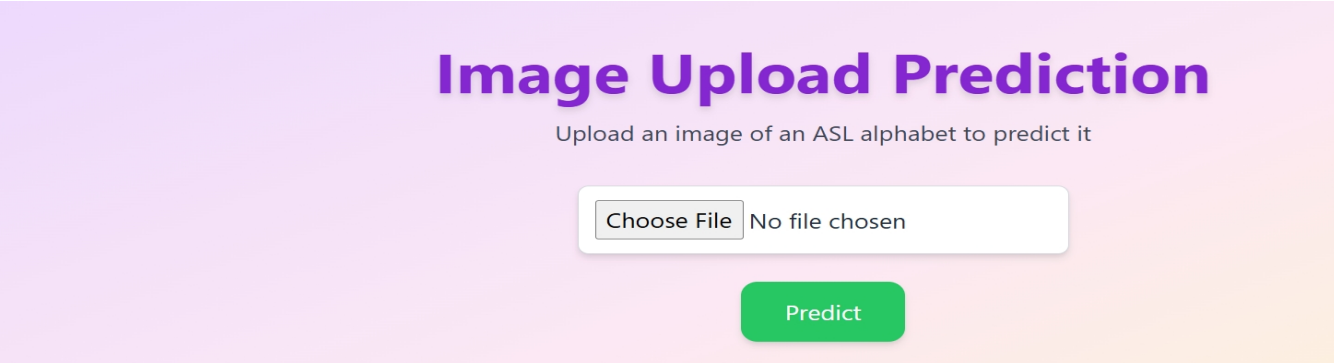
RESULTS AND DISCUSSION:

1. Open terminal in the same directory of the model and run the command "**python application.py".**Then the preferred web browser will be opened and the flask app will be displayed.

```
(tf_env) C:\Users\thula\Documents\Mini_Project>python application.py
 * Serving Flask app 'application'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 137-659-271
```

2.We can Predict by uploading images or by using Real-Time Webcam.

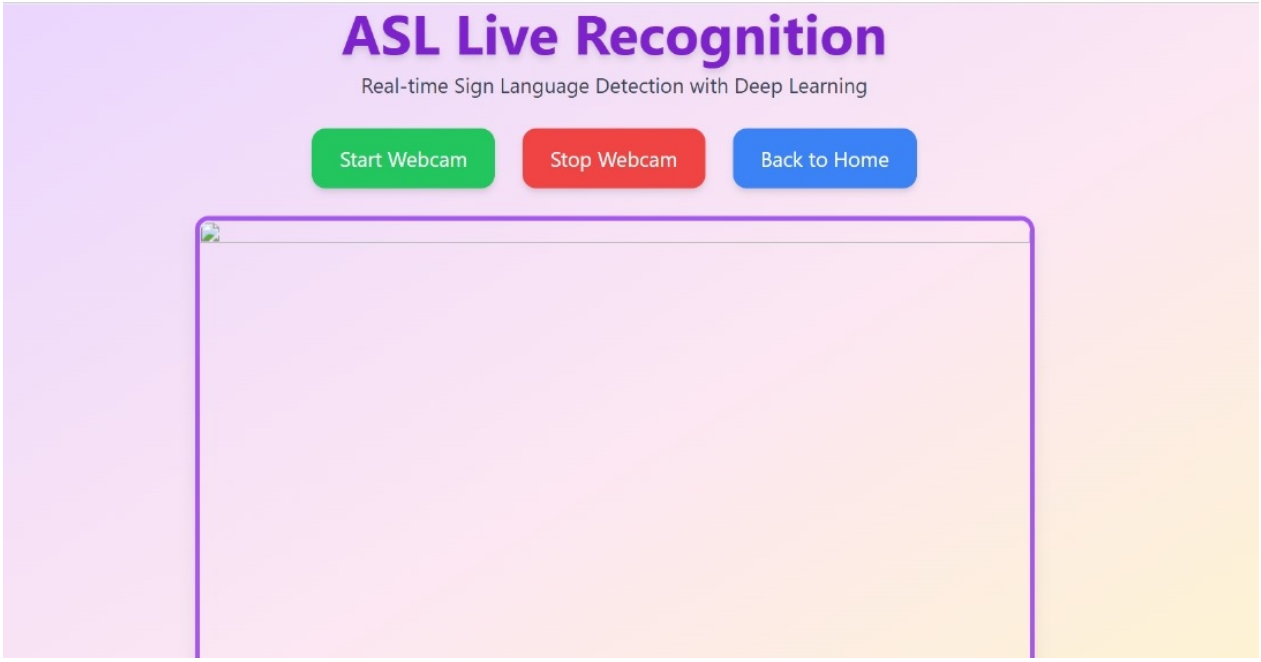
3. Image Upload prediction : Choose an Image and click on Predict Button



4.Real-Time prediction by using Webcam:

BY clicking on Start Webcam, we can start live Webcam which can predict our hand signs

By clicking on Stop Webcam,we can stop live prediction



RESULTS :

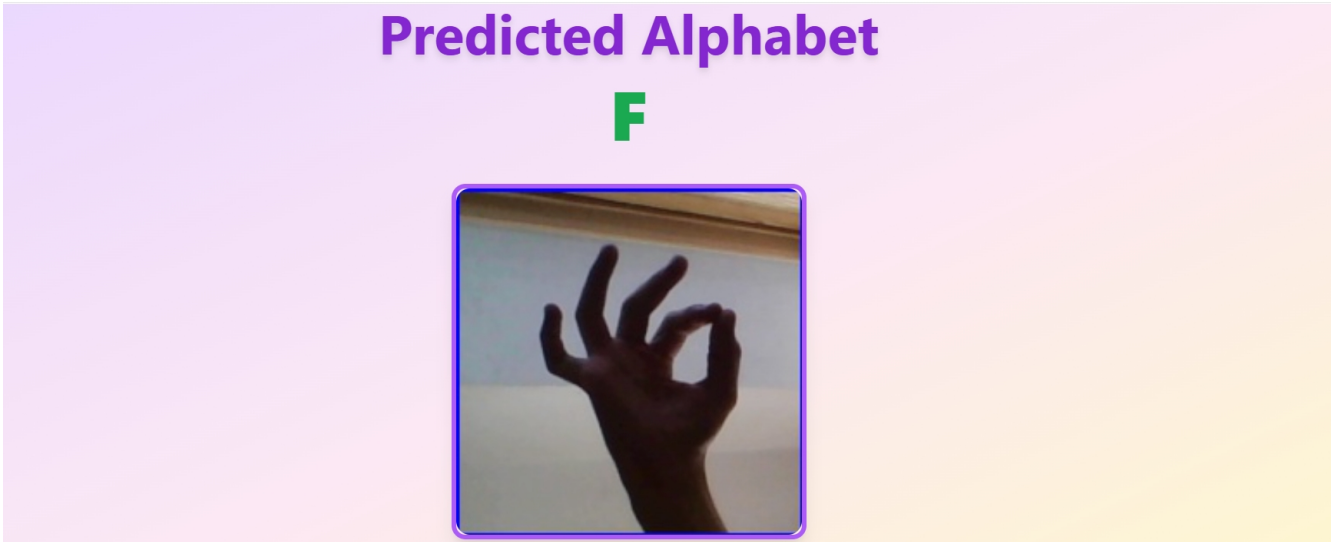


Figure : Output by using Image Upload prediction

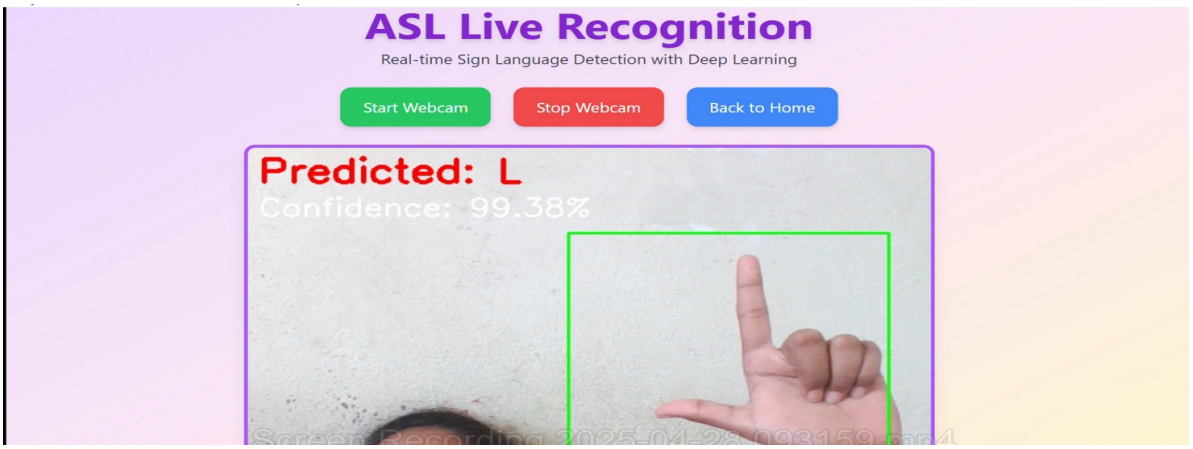


Figure: Output by using Real-Time prediction

CHAPTER-7

CONCLUSION:

This project successfully demonstrates the development of an American Sign Language (ASL) alphabet recognition system using Convolutional Neural Networks (CNNs). By training the model on a labeled dataset of ASL hand gestures, the system can accurately classify static images into one of 26 English alphabet categories. The integration of this model into a Flask web application further enhances its usability by offering both image upload and real-time webcam prediction interfaces. Overall, the project contributes to assistive technology, aiming to bridge communication gaps between the Deaf and hearing communities.

Future Work

1.Full Word and Sentence Recognition:

Extend the system from single-letter recognition to full ASL words and continuous gesture sentences using Recurrent Neural Networks (RNNs) or Transformers.

2. Dynamic Gesture Detection:

Incorporate temporal data handling (e.g., using LSTM or 3D CNNs) to recognize dynamic signs and motion-based gestures.

3.Multi-Hand and Contextual Recognition:

Improve the model to handle gestures involving both hands and incorporate facial expressions for more accurate interpretation.

4.Mobile and Edge Deployment:

Optimize the model for deployment on mobile devices or embedded systems for real-time accessibility in offline environments.

5.Multilingual Sign Language Support

Expand the system to support other sign languages such as British Sign Language (BSL) or Indian Sign Language (ISL) to increase inclusivity.

References:

1. TensorFlow Documentation

TensorFlow: An end-to-end open-source platform for machine learning.

Available at: https://www.tensorflow.org/

2. Keras API Reference

Keras: Deep learning API, running on top of TensorFlow.

Available at: https://keras.io/

3. American Sign Language (ASL) Information

American Sign Language Overview by National Institute on Deafness and Other Communication Disorders (NIDCD).

Available at: https://www.nidcd.nih.gov/health/american-sign-language

4. Flask Documentation

Flask: A lightweight WSGI web application framework.

Available at: https://flask.palletsprojects.com/

5. OpenCV Documentation

OpenCV: An open-source computer vision and machine learning software library.

Available at: https://opencv.org/

Additional Academic References:

•S. Pigou, S. Dieleman, P. Kindermans, and B. Schrauwen, "Sign Language Recognition Using Temporal Classification," Lecture Notes in Computer Science,vol. 9303, pp. 572–578, 2015.

•J. Huang, W. Zhou, Q. Zhang, and H. Li, "Video-Based Sign Language Recognition Without Temporal Segmentation," AAAI Conference on Artificial Intelligence, 2018.

•R. Rastgoo, K. Sarrafzadeh, and J. Escalera, "Video-based isolated hand sign language recognition using a deep cascaded model," Multimedia Tools and Applications, 2020.

•A. Kumar, S. Sharma, and R. Bharti, "A Deep Learning Approach to American Sign Language Recognition," International Journal of Computing and Digital Systems, 2022.

•J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.