ASL ALPHABET IMAGE RECOGNITION

AN INDUSTRY ORIENTED MINI REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

MANGU SATHVIKA	**20UK1A6614**
CHILUMULA DINESH	**20UK1A6643**
SREERAMOJU AKSHAY	**20UK1A6606**
VENNAPU RAMU	**20UK1A6628**

Under the guidance of

Mr. B. Naresh

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION
INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled "ASL ALPHABET IMAGE RECOGNITION" is being submitted by MANGU SATHVIKA (20UK1A6614) ,CHILUMULA DINESH (20UK1A6643),SREERAMOJU AKSHAY(20UK1A6651) ,VENNAPU RAMU (20UK1A6628) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023- 2024.

Project Guide
Mr.B.Naresh
(Assistant Professor)

HOD
Dr. K. Sharmila
(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr.P.PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr.K.SHARMILA**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **B.Naresh,** Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

MANGU SATHVKA (20UK1A6614)
CHILUMULA DINESH (20UK1A6643)
SREERAMOJU AKSHAY (20UK1A6606)
VENNAPU RAMU (20UK1A6628)

ABSTRACT

In this discussion, we outline the key steps involved in creating an image recognition system for the American Sign Language (ASL) alphabet. We highlight the importance of data collection, preprocessing, and labeling while emphasizing the need for proper data splitting. Feature extraction and model selection, with a focus on convolutional neural networks (CNNs), are crucial for the success of such a system. Training, validation, and testing phases are detailed, leading to potential deployment in real-time ASL alphabet recognition. We also provide essential tools and libraries for the project, including Python, OpenCV, TensorFlow, and PyTorch , and stress the significance of ethical considerations in creating inclusive and respectful ASL recognition solutions.

TABLE OF CONTENTS:-

1.INTRODUCTION

1.1.OVERVIEW

Creating an image recognition system for the American Sign Language (ASL) alphabet is a multifaceted endeavor. The process entails several critical stages. It begins with data collection, where a dataset of images representing ASL alphabet letters is amassed, either through existing sources or by capturing real-world signers. Subsequently, data preprocessing is applied to standardize and enhance image quality. Each image must be meticulously labeled with its corresponding ASL letter to facilitate supervised model training. The dataset is then divided into training, validation, and test subsets, a pivotal step in evaluating model performance. Features are extracted from the images to represent the unique signs, with convolutional neural networks (CNNs) often playing a pivotal role in this feature learning process. Model selection is a key decision, with CNNs again emerging as a popular choice due to their capacity for spatial feature comprehension. Training the model follows, optimizing parameters to minimize prediction errors, and the validation stage assesses its proficiency and guards against overfitting. Ultimately, testing on a dedicated dataset gauges its real-world performance. Successful models may find application in real-time ASL alphabet recognition, possibly integrated into mobile apps or other platforms. This endeavor is underscored by a commitment to data quality, robust model selection, and a deep consideration of ethical and inclusive practices, ensuring that ASL recognition systems are both accurate and respectful of the Deaf and hard of hearing community.

1.2.PURPOSE

The purpose of developing an image recognition system for the American Sign Language (ASL) alphabet can be summarized in five key points:

1. **Enhancing Accessibility:** The primary goal is to make digital content and communication more accessible for the Deaf and hard of hearing community by providing a tool that can interpret and translate ASL alphabet signs.
2. **Inclusivity:** We aim to create an inclusive society where individuals who use ASL can engage more effectively in everyday life, from education and employment to social interactions and online communication.

3. **Empowering ASL Users:** This technology empowers ASL users to interact with digital platforms, thus reducing communication barriers and promoting independence.
4. **Versatile Applications:** The ASL recognition system can be integrated into various applications, such as educational tools, assistive devices, and communication platforms, making it a versatile solution for diverse needs.
5. **Cultural and Linguistic Respect:** The development of this technology is driven by a commitment to respecting the cultural and linguistic significance of ASL, ensuring that it aligns with the values and identity of the Deaf and hard of hearing community

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

- The existing problem in the domain of American Sign Language (ASL) alphabet recognition is twofold. Firstly, it revolves around the substantial communication barriers faced by the Deaf and hard of hearing community, especially in digital settings, where ASL often goes uninterpreted.
- Existing automated systems for ASL sign recognition frequently fall short in delivering accurate and reliable interpretations, with issues arising from variations in signing styles and environmental conditions. These limitations restrict accessibility, impeding full participation in education, employment, and social interactions for ASL users.
- Secondly, the scarcity of comprehensive ASL datasets poses a significant challenge for the development of robust recognition models. Moreover, there's an ethical imperative to ensure that ASL recognition technology not only functions effectively but also respects the cultural and linguistic nuances inherent to ASL.
- Addressing these issues is crucial to improving accessibility and inclusivity for the Deaf and hard of hearing community and fostering the development of accurate, culturally sensitive ASL recognition systems.

2.2 PROPOSED SOLLUTION

A proposed solution to address the existing problems in American Sign Language (ASL) alphabet recognition involves the application of deep learning concepts, specifically leveraging Convolutional Neural Networks (CNNs). CNNs are well-suited for image analysis, making them a valuable tool for ASL recognition. Here's how this solution could work:

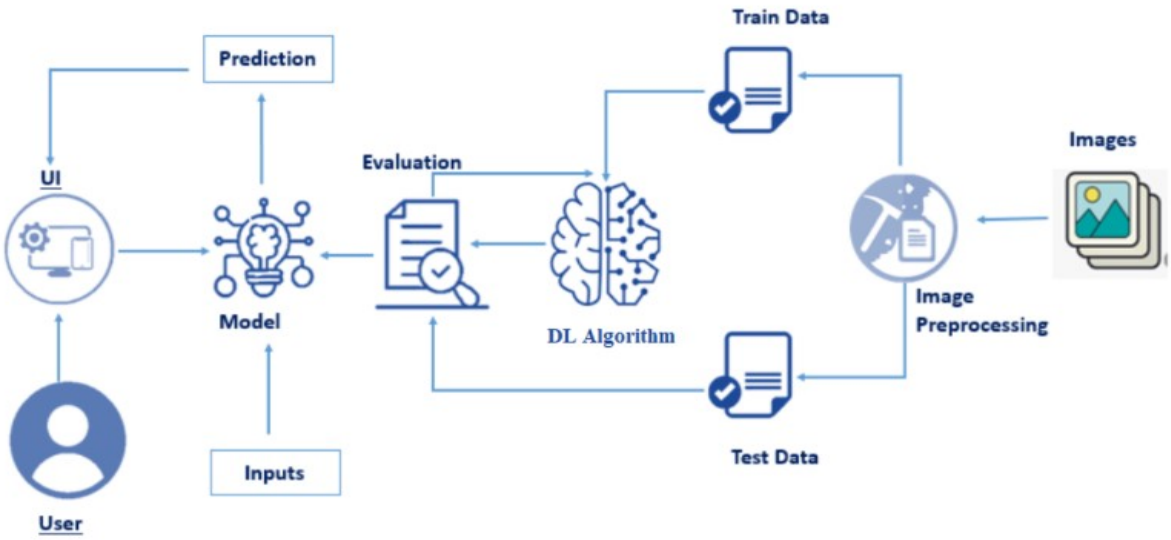
Leveraging CNNs for ASL Alphabet Recognition:

By harnessing the power of CNNs, we can significantly improve the accuracy and robustness of ASL alphabet recognition systems. Here's how:

1. **Data Augmentation:** To mitigate the issue of limited ASL datasets, data augmentation techniques can be applied to generate diverse training examples. This helps the model become more robust and adaptable to various signing styles and conditions.
2. **Feature Learning:** CNNs are designed to automatically learn hierarchical features from images. This is particularly advantageous in ASL recognition, as it allows the model to extract relevant spatial information from sign images.
3. **Real-time Processing:** CNNs can be optimized for real-time processing, making them suitable for applications like live interpretation and ASL communication on digital platforms.
4. **Model Fine-tuning:** Transfer learning can be employed by fine-tuning pre-trained CNN models, which have already learned valuable features from massive image datasets. Fine-tuning with ASL-specific data can significantly improve recognition performance.
5. **Ethical Considerations:** When developing an ASL recognition system based on CNNs, it is crucial to integrate ethical considerations. This includes respecting the cultural and linguistic nuances of ASL and ensuring that the technology is inclusive and respectful of the Deaf and hard of hearing community.

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection: Collect or download the dataset that you want to train your CNN on.
- Data Preprocessing: Preprocess the data by resizing, normalizing, and splitting the data into training and testing sets.
- Model Building:

  a. Import the necessary libraries for building the CNN model

  b. Define the input shape of the image data

  c. Add layers to the model:

  i. Convolutional Layers: Apply filters to the input image to create feature maps

  ii. Pooling Layers: Reduce the spatial dimensions of the feature maps

iii. Fully Connected Layers: Flatten the output of the convolutional layers and apply fully connected layers to classify the images

d. Compile the model by specifying the optimizer, loss function, and metrics to be used during training

- Model Training: Train the model using the training set with the help of the ImageDataGenerator class to augment the images during training. Monitor the accuracy of the model on the validation set to avoid overfitting.
- Model Evaluation: Evaluate the performance of the trained model on the testing set. Calculate the accuracy and other metrics to assess the model's performance.
- Model Deployment: Save the model for future use and deploy it in real-world applications.

4.EXPERIMENTAL INVESTIGATION

Data set information :

This data set consists of a set of 870 images. Each image contains a hand making the shape of an ASL letter (with some variation). The purpose of this data set is to act as a sort of validation set to see how good the preprocessing and the model are, based on the following ASL Alphabet data set also available on Kaggle: https://www.kaggle.com/grassknoted/asl-alphabet.

Content:

There are 30 images for each symbol, A-Z, delete, space, and nothing, making 870 images in total. The images are 200x200 8-bit photos to match the asl-alphabet data set, and are organized in a folder structure to make it easy to use `flow_from_directory` in Keras.

set up the Kaggle API credentials.

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

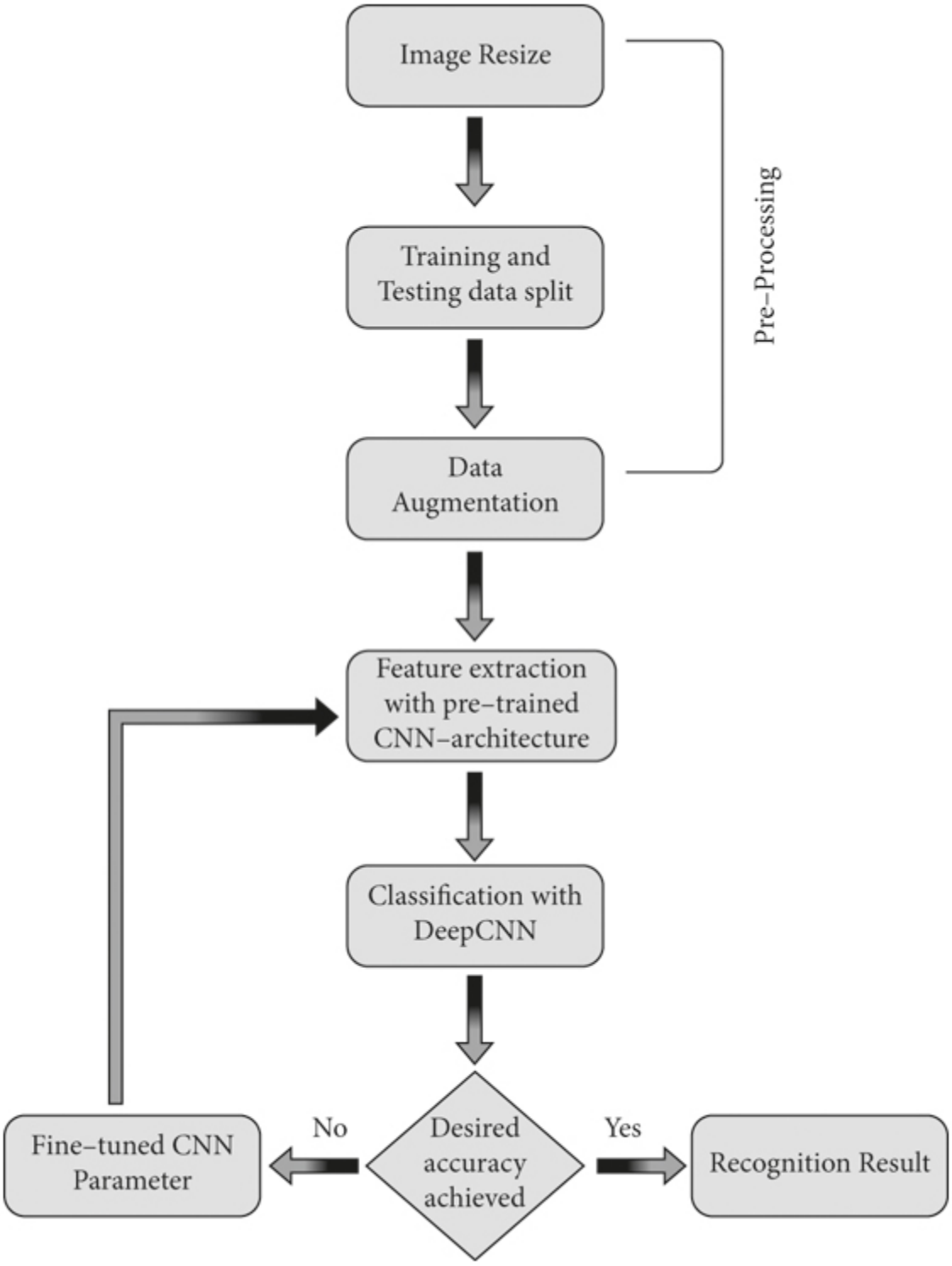
Download and unzip dataset

```
[ ]  !kaggle datasets download -d grassknoted/asl-alphabet
```

```
Downloading asl-alphabet.zip to /content
100% 1.02G/1.03G [00:46<00:00, 25.5MB/s]
100% 1.03G/1.03G [00:46<00:00, 23.8MB/s]
```

```
[ ]  !unzip asl-alphabet.zip -d asl-alphabet
```

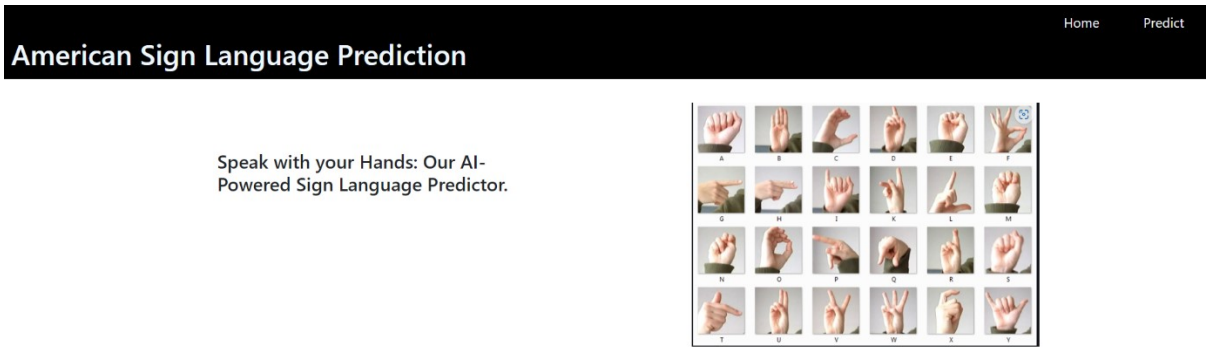
5.FLOWCHART



6.RESULT

Web interface images

Home.html




```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>American Sign Language Prediction</title>
    <!-- Add Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <!-- Add Bootstrap JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.bundle.min.js"></script>
    <link rel="stylesheet" href="/static/style.css">

</head>

<body>
    </style>
    <header class="header">
        <div class="topnav">
            <a href="/predict" style="color: aliceblue;">Predict</a>
            <a href="/home" style="color: aliceblue;">Home</a>
        </div>
        <div class="header-content">
            <h1>American Sign Language Prediction</h1>
        </div>
    </header>
    <div class="container">
        <div class="left-column">
            <!-- Add your content here on the left side -->
            <h4>Speak with your Hands: Our AI-Powered Sign Language Predictor.</h4>
        </div>
        <div class="right-column">
            <!-- Add your image on the right side -->
            <img src="/static/img.png" alt="Image">
        </div>
    </div>
</body>
</html>
```

Predict.html


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>American Sign Language Prediction</title>
    <!-- Add Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <!-- Add Bootstrap JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.bundle.min.js"></script>
    <link rel="stylesheet" href="/static/style1.css">
</head>
<body>
    <header class="header">
        <div class="topnav">
            <a href="/templates/predict.html" style="color: aliceblue;">Predict</a>
            <a href="/templates/home.html" style="color: aliceblue;">Home</a>
        </div>
        <div class="header-content">
            <h1>American Sign Language Prediction</h1>
        </div>
    </header>
    <div class="content">
        <div class="form-container">
            <form id="image-form" method="post" action="/predict" enctype="multipart/form-data">
                <input type="file" id="file_upload" name="image" accept="image/*"><br><br>
                <input type="submit" value="Submit">
            </form><br>
            <img id="image-preview" style="border-radius: 5px;"/>
        </div><br><br>
        <h1 style="color: black;">The hand sign represents:</h1><br>
        <div id="prediction-result">
            {% if image %}
            <img src="{{ image_path }}" alt="Uploaded Image">
        {% endif %}
            {% if pred %}
                <p>Predicted Sign: {{ pred }}</p>
            {% endif %}
        </div>
    </div>
    <script src="/static/script.js"></script>
</body>
</html>
```

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Enhanced Accuracy:** CNNs excel at capturing spatial features in images, leading to significantly improved recognition accuracy for ASL signs. Their ability to automatically learn hierarchical features enables the model to better understand the nuances of different signs and signing styles.
2. **Robustness to Variability:** CNNs can handle variations in lighting conditions, background clutter, and hand positioning, making them robust to real-world challenges in ASL recognition. This adaptability ensures reliable performance across diverse scenarios.
3. **Real-time Processing:** CNNs can be optimized for real-time processing, making them suitable for applications that require live interpretation of ASL signs. This capability is crucial for seamless communication and interaction on digital platforms.
4. **Transfer Learning:** Leveraging pre-trained CNN models and fine-tuning them with ASL-specific data can expedite the development process. Transfer learning enhances recognition performance, especially when dealing with limited ASL datasets.
5. **Inclusive Technology:** CNN-based ASL recognition systems, when developed with ethical considerations, respect the cultural and linguistic nuances of ASL, promoting inclusivity and accessibility for the Deaf and hard of hearing community. By facilitating effective communication, these systems empower ASL users to engage more fully in education, employment, and social interactions.

The advantages of using CNNs in ASL recognition systems contribute to more accurate, robust, and inclusive technology that can improve the quality of life for the Deaf and hard of hearing community.

DISADVANTAGES:

1. **Data Requirements:** Training CNNs effectively requires large and diverse datasets. For ASL recognition, collecting a comprehensive dataset with diverse signing styles and variations can be challenging, leading to potential limitations in model generalization.

2. **Complexity and Resource Intensiveness:** CNNs are computationally intensive and require substantial resources, including powerful GPUs, for training and inference. This can be a barrier for small-scale projects or applications with limited computational resources.
3. **Overfitting:** Overfitting, a common problem in deep learning, occurs when a model becomes too specialized in the training data and may not generalize well to unseen data. Effective regularization techniques are needed to mitigate this issue.
4. **Interpretability:** CNNs are often considered "black box" models, making it challenging to interpret why they make certain predictions. In applications like ASL recognition, interpretability is important for understanding errors and improving the system's reliability.
5. **Cultural Sensitivity:** While CNNs can enhance recognition accuracy, they may not inherently respect the cultural and linguistic aspects of ASL. Ensuring that the technology aligns with the values and identity of the Deaf and hard of hearing community while maintaining inclusivity is a complex challenge.

8.APPLICATIONS

1. **Accessibility Tools:** ASL recognition systems integrated into smartphones, tablets, and computers can serve as accessibility tools for the Deaf and hard of hearing community. They enable real-time sign language interpretation, facilitating effective communication in various settings, including education and customer service.
2. **Educational Resources:** ASL recognition systems can be integrated into educational platforms to provide interactive and personalized learning experiences. These systems can offer immediate feedback and support to ASL learners, helping them improve their signing skills.
3. **Assistive Devices:** Wearable devices equipped with ASL recognition technology can assist Deaf individuals in their daily lives. For example, these devices can translate sign language into text or speech, making it easier to communicate with people who do not know ASL.

4. **Communication Apps:** ASL recognition can be incorporated into video conferencing and messaging applications, allowing Deaf individuals to communicate via sign language with others. This is especially valuable for remote work, social interactions, and telehealth services.
5. **Sign Language Learning Games:** Interactive games and applications that use ASL recognition can engage users in fun and educational experiences. These platforms can help individuals practice and reinforce their ASL skills in an enjoyable manner.

9.CONCLUSION

- In conclusion, the development and implementation of American Sign Language (ASL) alphabet recognition systems, particularly those based on Convolutional Neural Networks (CNNs), hold great promise for improving the lives of the Deaf and hard of hearing community. These systems address significant challenges, such as communication barriers and limited accessibility, by providing accurate and real-time interpretation of ASL signs. By harnessing the power of CNNs, these recognition systems offer enhanced accuracy, robustness to variations, and adaptability to real-world scenarios.
- However, it's important to acknowledge the challenges that come with the technology, including data requirements, complexity, overfitting, and the need for interpretability. Ethical considerations are paramount to ensure that ASL recognition systems respect the cultural and linguistic nuances of ASL and maintain inclusivity.
- The applications of these systems are wide-ranging, from accessibility tools and educational resources to assistive devices and communication apps. They contribute to a more inclusive society, empowering Deaf individuals to participate fully in education, employment, and social interactions. Ultimately, the development of ASL recognition technology aligns with the values of equality, understanding, and engagement, making strides towards a more inclusive and accessible world for all, regardless of their hearing abilities.

10.FUTURE SCOPE

1. **Multi-modal Recognition:** The future holds the potential for multi-modal recognition systems that can interpret not only ASL signs but also facial expressions and body language to provide a

more comprehensive understanding of signed communication. This holistic approach will enhance the accuracy and richness of ASL recognition.

2. **Sign Language Dialects:** ASL recognition systems may evolve to recognize different sign language dialects and regional variations. This expansion will be vital in accommodating the diversity of signing styles and regional linguistic nuances.
3. **Gesture-Based Interfaces:** ASL recognition technology can extend beyond ASL into recognizing a broader range of gestures and non-verbal communication. This opens doors to applications in virtual reality, augmented reality, and human-computer interaction.
4. **Wearable Technology:** The integration of ASL recognition into wearable devices will become more prevalent. These devices will provide on-the-go support for Deaf individuals, assisting in daily interactions, navigation, and accessibility.
5. **Improved Ethical Considerations:** The future of ASL recognition systems will place an even greater emphasis on ethical and inclusive design. Ensuring that these technologies respect ASL culture and the Deaf community's values will be a central focus, leading to more culturally sensitive and respectful systems.

11.BIBILOGRAPHY

- H. S. Hasan and S. A. Kareem, “Human computer interaction for vision based hand gesture recognition: a survey,” in *Proceedings of the 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, pp. 55–60, IEEE, Kuala Lumpur, Malaysia, November 2012.

- S. Y. Kim, H. G. Han, J. W. Kim, S. Lee, and T. W. Kim, “A hand gesture recognition sensor using reflected impulses,” *IEEE Sensors Journal*, vol. 17, no. 10, pp. 2975-2976, 2017.

- Y. Cui and J. Weng, “A learning-based prediction-and-verification segmentation scheme for hand sign image sequence,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 798–804, 1999.

- F. Shahzad, A. R. Javed, Y. B. Zikria, R. Su, and Z. Jalil, “Future smart cities: requirements, emerging technologies, applications, challenges, and future aspects,” *TechRxiv*, 2021.

- A. Kumar and A. Kumar, “Dog breed classifier for facial recognition using convolutional neural networks,” in *Proceedings of the 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 508–513, IEEE, Thoothukudi, India, December 2020.

- T. Reddy Gadekallu, G. Srivastava, and M. Liyanage, “Hand gesture recognition based on a Harris hawks optimized convolution neural network,” *Computers & Electrical Engineering*, vol. 100, Article ID 107836, 2022.

- H. Ben Amara, “End-to-end Multiview Gesture Recognition for Autonomous Car Parking system,” University of Waterloo, Waterloo, Canada, 2019, Master’s Thesis.

- G. T. R. Chiranji Lal Chowdhary and B. D. Parameshachari, *Computer Vision and Recognition Systems: Research Innovations and Trends*, CRC Press, 2022.

- S. Gadamsetty, Ch Rupa, Ch Anusha, C. Iwendi, and T. Reddy Gadekallu, “Hash-based deep learning approach for remote sensing satellite imagery detection,” *Water*, vol. 14, no. 5, p. 707, 2022.

- R. U. Khan, X. Zhang, M. Alazab, and R. Kumar, “An improved convolutional neural network model for intrusion detection in networks,” in *Proceedings of the 2019 Cybersecurity and Cyberforensics Conference (CCC)*, pp. 74–77, IEEE, Melbourne, VIC, Australia, May 2019.

- G. Plouffe and A. M. Cretu, “Static and dynamic hand gesture recognition in depth data using dynamic time warping,” *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 2, pp.

- S. Nisar and M. Tariq, “Dialect recognition for low resource language using an adaptive filter bank,” *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 16, no. 04, Article ID 1850031, 2018.

11.APPENDIX

Model building

1)Dataset
2)Google colab and VS code Application Building

1. HTML file (Index file, Predict file)
1. CSS file
2. Models in pickle format

SOURCE CODE:

HTML CODE

Home.html

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: #333;
  color: white;
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
  padding: 0;
}

.container {
  width: 30%;
  padding: 10px 20px;
}
```

```
h1 {
  text-align: center;
}
```

```
form {
  background-color: #000;
  padding: 20px;
  color: white;
  width: 40%;
  box-sizing: border-box;
}
```

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: 2px solid rgb(33, 33, 35);
  border-radius: 4px;
}
```

```
input[type=text]:hover {
  background-color: white;
}
```

```
input[type=submit] {
  background-color: blue;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
```

```
  width: 100%; /* Make the submit button same width as input values */
}
```

```
input[type=submit]:hover {
  background-color: rgb(57, 190, 207);
}
</style>
</head>
<body>
```

```
 </div>
        <div class="header-content">
            <h1>American Sign Language Prediction</h1>
        </div>
    </header>
    <div class="container">
        <div class="left-column">
            <!-- Add your content here on the left side -->
            <h4>Speak with your Hands: Our AI-Powered Sign Language Predictor.</h4>
        </div>
        <div class="right-column">
<div class="right-column">
            <!-- Add your image on the right side -->
            <img src="/static/img.png" alt="Image">
        </div>
    </div>
```

```
</body>
</html>
```

Predict.html

```
<!DOCTYPE html>
<html>

<head>
    <style>
        /* Add your CSS styles here */
        body {
            font-family: Arial, sans-serif;
            background-color: #c0c0c0; /* Set background color to ash */
            margin: 0;
            padding: 0;
        }
```

```
        /* Header styles */
        .main_1 {
            background-color: #007BFF;
            color: white;
            padding: 10px 0;
            text-align: center;
        }
```

```css
.head1 {
    font-size: 24px;
    margin: 0;
}
```

```css
.style_but2 {
    background-color: #0056b3;
    border: none;
    padding: 10px 20px;
    margin: 0 10px;
    border-radius: 5px;
    text-decoration: none;
    color: white;
    cursor: pointer;
    font-weight: bold;
}
```

```css
.style_but2 a {
    color: white;
    text-decoration: none;
}
```

```css
.style_but2:hover {
    background-color: #003d80;
}
```

```css
/* Left section styles */
.main_3 {
    display: flex;
    justify-content: space-between;
    padding: 20px;
    background-color: #fff;
    border-radius: 5px;
    margin: 20px;
}
```

```css
.main_3 img {
    max-width: 100%;
    height: auto;
}
```

```css
.left-input-container {
    width: 30%;
}
```

```css
/* Right section styles */
.second_main {
    display: flex;
    justify-content: space-between;
    padding: 20px;
    background-color: #fff;
    border-radius: 5px;
    margin: 20px;
}
```

```
        .para {
            font-weight: bold;
        }
```

```
        label {
            display: block;
            margin-top: 10px;
        }
```

```
        .wide-input {
            width: 70%;
            padding: 10px;
            margin-top: 5px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
```

```
        .style_but {
            background-color: #007BFF;
            border: none;
            padding: 10px 20px;
            margin: 10px 0;
            border-radius: 5px;
            color: white;
            cursor: pointer;
            font-weight: bold;
        }
```

```
        .style_but:hover {
            background-color: #0056b3;
        }
```

```
        /* Predictions section styles */
        .predictions {
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
            margin: 20px;
            background-color: #f9f9f9;
        }
```

```
        /* Set the background color of container1 to ash and move it slightly to the right */
        .container1 {
            background-color: #c0c0c0;
            margin: 0 100px; /* Move container1 slightly to the right */
            max-width: 600px; /* Set a maximum width for the container */
            padding: 50px;
            border-radius: 5px;
        }
    </style>
</head>
```

```
<body>
 <div class="header-content">
            <h1>American Sign Language Prediction</h1>
        </div>
```

```
    </header>
    <div class="content">
        <div class="form-container">
            <form id="image-form" method="post" action="/predict" enctype="multipart/form-data">
                <input type="file" id="file_upload" name="image" accept="image/*"><br><br>
                <input type="submit" value="Submit">
            </form><br>
            <img id="image-preview" style="border-radius: 5px;"/>
        </div><br><br>
        <h1 style="color: black;">The hand sign represents:</h1><br>
        <div id="prediction-result">
            {% if image %}
            <img src="{{ image_path }}" alt="Uploaded Image">
        {% endif %}
            {% if pred %}
                <p>Predicted Sign: {{ pred }}</p>
            {% endif %}
        </div>
    </div>
```

```
    <div class="container1">
```

```
</html>
```

FLASK CODE

```
import numpy as np
from PIL import Image
import os
import tensorflow as tf
from tensorflow.keras.models import load_model
from flask import Flask, render_template, request

app = Flask(__name__)
```

```
class_labels = {
    0: "COVID-19",
    1: "Lung Capacity",
    2: "Normal",
    3: "Viral Pneumonia"
}
```

```
# Load the sign language alphabet prediction model
sign_language_model = load_model("asl_vgg16_best_weights.h5", compile=False)
sign_language_labels = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

```
def preprocess_image(image_path):
    img = Image.open(image_path)
    img = img.resize((64, 64))
    img = np.array(img)
    img = tf.keras.applications.mobilenet_v2.preprocess_input(img)
```

```
    return img

@app.route("/")
def home():
    return render_template('home.html')

@app.route("/home")
def home1():
    return render_template('home.html')

@app.route("/predict", methods=['GET', 'POST'])
def predict():
    sign_language_prediction = None
    image_path = None

    if request.method == 'POST' and 'image' in request.files:
        try:
            f = request.files['image']
            basepath = os.path.dirname(__file__)
            filepath = os.path.join(basepath, 'uploads', f.filename)
            f.save(filepath)

            img = preprocess_image(filepath)
            predictions = sign_language_model.predict(np.array([img]))

            if len(predictions) > 0:
                max_prediction_index = np.argmax(predictions)
                if 0 <= max_prediction_index < len(sign_language_labels):
                    sign_language_prediction = sign_language_labels[max_prediction_index]
                else:
                    sign_language_prediction = "Nothing or Space"
            else:
                sign_language_prediction = "No Predictions Available"

            image_path = '/uploads/' + f.filename  # Adjust the path as needed
        except Exception as e:
            print("An error occurred:", str(e))

    return render_template('predict.html', pred=sign_language_prediction, image_path=image_path)


if __name__ == "__main__":
    app.run(debug=True)
```

CODE SNIPPETS

Data Preparation

```
import numpy as np
import pandas as pd
import gc
import string
import time
import random
import imutils
from PIL import Image
from tqdm import tqdm
tqdm.pandas()

# Visualization
import matplotlib
import matplotlib.pyplot as plt
import plotly
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
from sklearn.manifold import TSNE

# Model
from sklearn.model_selection import train_test_split
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_ar
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout
from keras.models import load_model, Model
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

```
[ ]  # Configuration
     class CFG:
         batch_size = 64
         img_height = 64
         img_width = 64
         epochs = 10
         num_classes = 29
         img_channels = 3

     def seed_everything(seed: int):
         random.seed(seed)
         os.environ["PYTHONHASHSEED"] = str(seed)
         np.random.seed(seed)
         tf.random.set_seed(seed)
```

```
[ ]  #Create Metadata
list_path = []
list_labels = []
for label in labels:
    label_path = os.path.join(TRAIN_PATH, label, "*")
    image_files = glob.glob(label_path)

    sign_label = [label] * len(image_files)

    list_path.extend(image_files)
    list_labels.extend(sign_label)

metadata = pd.DataFrame({
    "image_path": list_path,
    "label": list_labels
})

metadata
```

Data Preprocessing

```
# Labels
TRAIN_PATH = "asl-alphabet/asl_alphabet_train/asl_alphabet_train"
labels = []
alphabet = list(string.ascii_uppercase)
labels.extend(alphabet)
labels.extend(["del", "nothing", "space"])
print(labels)
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N'
```

```
# Split Dataset to Train 0.7, Val 0.15, and Test 0.15
X_train, X_test, y_train, y_test = train_test_split(
    metadata["image_path"], metadata["label"],
    test_size=0.15,
    random_state=2023,
    shuffle=True,
    stratify=metadata["label"]
)
data_train = pd.DataFrame({
    "image_path": X_train,
    "label": y_train
})

X_train, X_val, y_train, y_val = train_test_split(
    data_train["image_path"], data_train["label"],
    test_size=0.15/0.70,
    random_state=2023,
    shuffle=True,
    stratify=data_train["label"]
)
```

Data Agumentation

```
]  # Data Augmentation (Just Rescale)
   def data_augmentation():
       datagen = ImageDataGenerator(rescale=1/255.,)
       # Training Dataset
       train_generator = datagen.flow_from_dataframe(
           data_train,
           directory="./",
           x_col="image_path",
           y_col="label",
           class_mode="categorical",
           batch_size=CFG.batch_size,
           target_size=(CFG.img_height, CFG.img_width),
       )

       # Validation Dataset
       validation_generator = datagen.flow_from_dataframe(
           data_val,
           directory="./",
           x_col="image_path",
           y_col="label",
```

Model Traning

```
# Load VGG16 model and modify for ASL recognition
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(

for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(29, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

display(model.summary())
display(tf.keras.utils.plot_model(model, to_file='vgg16.png', show_shap
```

```
# Compile and train the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy',

# Callbacks
checkpoint = ModelCheckpoint('asl_vgg16_best_weights.h5', save_best_only=True, moni
```

```
# Train the Model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // CFG.batch_size,
    epochs=CFG.epochs,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // CFG.batch_size,
    callbacks=[checkpoint]
)
```

Model Evalution

```
scores = model.evaluate(test_generator)
print("%s: %.2f%%" % ("Evaluate Test Accuracy", scores[1]*100))
```

```
13050/13050 [==============================] - 84s 6ms/step - loss: 0.0380 - accurac
Evaluate Test Accuracy: 98.81%
```

```
# Confusion Matrix
fine_tuned_model = load_model("/content/asl_vgg16_best_weights.h5")
predictions = fine_tuned_model.predict(test_generator)

# Get the true labels from the generator
true_labels = test_generator.classes

# Compute the confusion matrix using tf.math.confusion_matrix
confusion_matrix = tf.math.confusion_matrix(
        labels=true_labels,
        predictions=predictions.argmax(axis=1),
        num_classes=29)
```

```
13050/13050 [==============================] - 60s 5ms/step
```

```
# Load the saved model
model = tf.keras.models.load_model('/content/asl_vgg16_best_weights.h5')
# Load the test image
image_path = '/content/asl-alphabet/asl_alphabet_test/asl_alphabet_test/A_test.jpg'
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (64, 64))

# Preprocess the image
img = tf.keras.applications.mobilenet_v2.preprocess_input(img)

# Make predictions on the image
predictions = model.predict(np.array([img]))

# Get the predicted class label
predicted_class = labels[np.argmax(predictions)]

print(f"The predicted class is {predicted_class}")
```

```
1/1 [==============================] - 0s 201ms/step
The predicted class is A
```

```
image_path = '/content/asl-alphabet/asl_alphabet_test/asl_alphabet_test/P_test.jpg'
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (64, 64))

# Preprocess the image
img = tf.keras.applications.mobilenet_v2.preprocess_input(img)

# Make predictions on the image
predictions = model.predict(np.array([img]))

# Get the predicted class label
predicted_class = labels[np.argmax(predictions)]

print(f"The predicted class is {predicted_class}")
```

```
WARNING:tensorflow:6 out of the last 2905 calls to <function Model.make_predict_function.<local
1/1 [==============================] - 0s 259ms/step
The predicted class is P
```