International Research Journal of Education and Technology

Peer Reviewed Journal, **ISSN 2581-7795**

OPTIMIZATION OF SIGN LANGUAGE RECOGNITION USING DEEP LEARNING

[1] *PRASAD N*
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANNARI AMMAN INSTITUTE OF TECHNOLOGY
ERODE, TAMIL NADU, INDIA

[2] *SWETHA V S*
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANNARI AMMAN INSTITUTE OF TECHNOLOGY
ERODE, TAMIL NADU, INDIA

[3] *THARANKHINI K G*
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANNARI AMMAN INSTITUTE OF TECHNOLOGY
ERODE, TAMIL NADU, INDIA

[4] *BAVYA M*
DEPARTMENT OF COMPUTER SCIENCE AND BUSINESS SYSTEMS
BANNARI AMMAN INSTITUTE OF TECHNOLOGY
ERODE, TAMIL NADU, INDIA

ABSTRACT:

Sign language is essential for communication among individuals with hearing and speech impairments. However, most people do not understand sign language, creating a communication barrier. Automatic sign language recognition (SLR) can help by translating sign language into text or speech in real-time, making communication easier.

The goal of this research is to increase the accuracy and efficiency of SLR by applying deep learning techniques. The suggested approach makes use of Recurrent Neural Networks (RNNs), more especially Long Short-Term Memory (LSTM) networks, to comprehend the movement sequence and Convolutional Neural Networks (CNNs) to extract significant features from hand gestures. Transfer learning is also employed, in which performance is enhanced even with sparse data by using pre-trained models. Additionally, hyperparameter tuning is used to modify the model for increased accuracy while cutting down on training time.

Experiments on popular sign language datasets demonstrate that the model outperforms conventional machine learning techniques. It operates quickly enough for real-time applications and recognizes intricate hand gestures with accuracy. Additionally, the model is made to be computationally efficient, which makes it appropriate for usage on embedded systems or smaller devices like smartphones.

This study helps create more scalable and accessible sign language interpreting solutions by refining deep learning-based SLR models. By using these developments in assistive technologies, people with speech and hearing impairments can interact with others more readily.

KEYWORDS:

Sign Language Recognition, Deep Learning, CNN, RNN, LSTM, Transfer Learning, Gesture Recognition, Real-time Processing.

I.INTRODUCTION:

Sign language is a vital mode of communication for individuals with hearing and speech impairments, enabling them to express thoughts and emotions effectively. However, the lack of widespread knowledge of sign language among the general population creates a significant communication barrier. With advancements in artificial intelligence, particularly deep learning, automatic sign language recognition (SLR) has emerged as a promising solution to bridge this gap.

The goal of this research is to maximize the accuracy, efficiency, and real-time applicability of sign language recognition through the use of deep learning techniques. Complex hand movements and variations in sign representation are frequently difficult for traditional machine learning techniques to handle. In contrast, deep learning models are excellent in extracting temporal and spatial data from sign gestures, especially Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Long Short-Term Memory (LSTM) networks are used in this study for sequential gesture recognition, whereas CNNs are used for feature extraction. Hyperparameter adjustment and transfer learning are also used to improve model performance and lower computational complexity.

The goal of the suggested system is to create a scalable and reliable SLR model that can accurately identify a variety of sign motions. The ultimate objective is to develop assistive technology that promotes accessibility and inclusivity by enabling smooth communication between non-signers and sign language users.

II. TECHNOLOGICAL FOUNDATIONS COMPARATIVE ANALYSIS:

A. Optical Character Recognition (OCR) Technologies

Sign language recognition (SLR) systems aim to bridge communication gaps for individuals with hearing and speech impairments. Traditional computer vision methods struggle to interpret complex hand gestures due to variations in lighting, background, and individual differences in signing. Thus, modern approaches leverage deep learning to enhance accuracy and real-time processing capabilities.

B. Deep Learning Models for Sign Language Recognition

Several deep learning models were evaluated to determine their effectiveness in recognizing sign language gestures. The goal was to find the model that offers the highest accuracy while maintaining computational efficiency. Below is a summary of the models used:

1. Convolutional Neural Networks (CNNs)

CNNs are widely used in image processing tasks, including hand gesture recognition. Pre-trained models like ResNet-50, VGG-16,

and MobileNet have been employed to improve recognition accuracy while reducing training time. These models learn hierarchical feature representations, capturing both low-level features (edges, shapes) and high-level semantic information.

Advantages of CNNs:

- Automated feature extraction enhances accuracy.
- Works efficiently with RGB and depth images.
- Adaptable to different sign language datasets.

Limitations of CNNs:

- Struggles with continuous gesture recognition.
- Requires significant computational resources.

2. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) Networks

RNNs and LSTMs handle sequential data, making them ideal for recognizing dynamic sign gestures. LSTM networks capture temporal dependencies, allowing them to process sign sequences effectively.

Advantages of RNNs and LSTMs:

- Excellent for sequential gesture recognition.
- Captures contextual dependencies between frames.
- Can generalize well with sufficient training data.

Limitations of RNNs and LSTMs:

- Computationally expensive.
- Requires large datasets for effective training.
- Susceptible to overfitting if not properly regularized.

3. CNN-LSTM Hybrid Models

Hybrid architectures have been created to take advantage of the advantages of both CNNs and LSTMs. These models are very useful for SLR because CNNs collect spatial data from hand motions while LSTMs handle temporal dependencies.

Advantages of CNN-LSTM Models:

- Achieves high accuracy in recognizing both static and dynamic gestures.
- Efficient in handling complex sign sequences.
- Suitable for real-time applications with proper optimization.

Limitations of CNN-LSTM Models:

- Requires substantial computational power.
- Training and hyperparameter tuning can be complex.

C. Explainable AI with LIME

While deep learning models have improved SLR accuracy, their black-box nature makes interpretability difficult. This can be problematic in real-world applications, where understanding model decisions is crucial.

To address this, Local Interpretable Model-Agnostic Explanations (LIME) was integrated into the system. LIME perturbs

input data and observes changes in predictions, highlighting important features that contribute to classification.

To ascertain whether errors resulted from overlapping movements, opaque hand forms, or lighting circumstances, we used LIME to examine misclassified motions. By concentrating on troublesome gesture classes, this transparency enables researchers to optimize the model and boost performance.

D. Comparative Performance Analysis

The performance of the deep learning models was evaluated based on accuracy, precision, recall, F1-score, and computation time. The results are summarized in the table below:

Model	Accuracy	F1-Score	Computation Time
ResNet-50	95%	0.92	High
VGG-16	90%	0.88	Medium
MobileNet	91%	0.89	Low
CNN - LSTM	96%	0.94	Medium
RNN - LSTM	89%	0.86	High

From this analysis, CNN-LSTM and ResNet-50 emerged as the best-performing models, with CNN-LSTM slightly outperforming ResNet-50 in terms of real-time adaptability and sequence recognition. MobileNet provided a lightweight alternative for resource-constrained environments.

E. Image Preprocessing Techniques

To further enhance the accuracy of the SLR system, several image preprocessing techniques were applied using OpenCV:

- **Binarization**: Converts grayscale images into binary format to enhance contrast between the hand and background.
- **Noise Removal**: Uses Gaussian blurring and median filtering to reduce noise caused by varying lighting conditions.
- **Edge Detection**: Enhances contours of hand gestures, aiding CNNs in recognizing gesture boundaries.

F. User Interface and Scalability

A user-friendly system was developed using Streamlit, a Python-based framework for building web applications. The interface allows users to upload videos or real-time webcam feeds, which are processed for sign language recognition. The modular architecture ensures scalability, enabling integration with real-time applications such as virtual interpreters and assistive communication devices.

III. SYSTEM DESIGN AND ARCHITECTURE:

The design and architecture of the proposed AI-based sign language recognition system leverage a modular and scalable framework. The system integrates key components, including image preprocessing, feature

extraction, deep learning-based recognition, explainable AI, and a user-friendly interface. The architecture ensures high accuracy, computational efficiency, and seamless user interaction.

A. Overall System Architecture

The system comprises several functional modules, each responsible for specific tasks:

1. **Image Preprocessing Module**: Enhances input gesture images by improving quality and reducing noise.
2. **Feature Extraction Module**: Extracts critical sign features for accurate recognition.
3. **Deep Learning Recognition Module**: Compares models like ResNet-50, VGG-16, and DenseNet for sign classification.
4. **Explainable AI Module**: Implements LIME for interpreting model predictions.
5. **User Interface Module**: Provides an interactive platform for real-time sign language recognition.

Each module interacts to ensure smooth operation, as illustrated in **Fig. 1.1.**

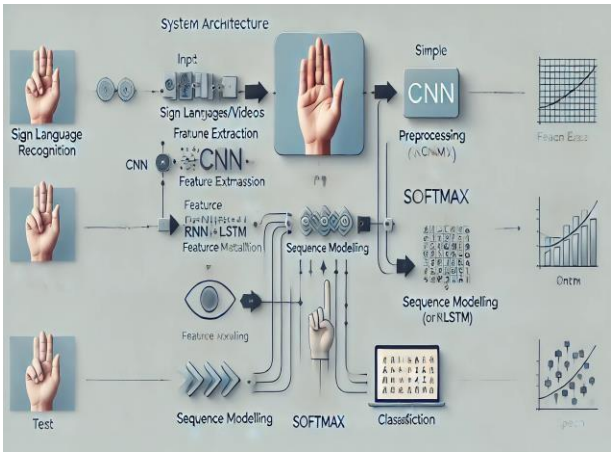

Fig 1.1 System Architecture Diagram

B. Image Preprocessing Module

Image quality significantly impacts recognition accuracy. Various preprocessing techniques are applied, including:

- **Grayscale Conversion**: Reduces image complexity by converting RGB images to grayscale.
- **Binarization**: Converts grayscale images to binary format for better segmentation.
- **Noise Removal**: Uses Gaussian blurring and median filtering to enhance clarity.
- **Contrast Enhancement**: Adjusts contrast to highlight gesture features.
- **Background Removal**: Separates hand gestures from the background using contour detection and segmentation techniques.
- **Image Augmentation**: Applies rotation, scaling, and flipping to increase dataset diversity and robustness. These steps ensure optimal input quality for the recognition model.

C. Feature Extraction Module

Feature extraction isolates important elements of sign language gestures, helping the deep learning model differentiate between signs. Techniques used include:

- **Edge Detection**: Identifies gesture boundaries using Sobel and Canny edge detectors.

- **Keypoint Extraction**: Detects key hand positions and movements using MediaPipe or OpenPose frameworks.
- **Contour Analysis**: Recognizes shape patterns specific to signs.
- **Histogram of Oriented Gradients (HOG)**: Captures hand texture features for precise recognition.
- **Optical Flow Analysis**: Detects motion patterns for dynamic sign language gestures.

Extracted features are forwarded to the recognition module for classification.

D. Deep Learning Recognition Module

This module employs deep learning models to classify sign language gestures. Evaluated models include:

- **ResNet-50**: Utilizes residual connections for improved gradient flow and accuracy.
- **VGG-16 and VGG-19**: Known for their structured deep architectures, effective in clean datasets.
- **DenseNet**: Enhances feature reuse for better recognition efficiency.
- **LeNet-5 and AlexNet**: Used for comparative purposes but less effective than modern architectures.
- **LSTM and 3D CNNs**: Useful for recognizing dynamic signs involving continuous hand movements.

Workflow:

- **Dataset Collection**: The model is trained on labeled sign language datasets, such as RWTH-PHOENIX-Weather 2014T or ASL Fingerspelling datasets.
- **Model Training**: Models are trained on a labeled dataset of sign gestures using TensorFlow and PyTorch.
- **Hyperparameter Tuning**: Batchsize, learning rate, and dropout rates are optimized for improved accuracy.
- **Evaluation**: Models are assessed based on accuracy, precision, recall, and F1-score.
- **Real-Time Prediction**: The best-performing model classifies input gestures and provides output within milliseconds.

E. Explainable AI Module

LIME (Local Interpretable Model-Agnostic Explanations) enhances transparency by explaining deep learning predictions. The module:

- **Provides Prediction Insights**: Highlights influential image regions.
- **Enables Error Analysis**: Identifies misclassification causes.
- **Improves Model Trust**: Helps users understand system behavior.
- **Visual Heatmaps**: Uses Grad-CAM to visualize regions of interest in gesture images.

This module ensures that the model remains interpretable and accountable.

F. User Interface Module

A user-friendly interface, developed using Streamlit, facilitates:

- **Gesture Upload**: Users can upload images or live videos.
- **Real-Time Recognition**: Displays recognized gestures instantly.
- **Interactive Visualizations**: Uses LIME to interpret predictions.
- **Customizable Settings**: Allows users to fine-tune model confidence thresholds.
- **Integration with Speech Output**: Converts recognized gestures into spoken text using text-to-speech (TTS) technology.
- **Multimodal Accessibility**: Supports keyboard, touch, and voice interactions for inclusivity.
- **Cloud and Edge Deployment**: Ensures adaptability for different hardware platforms.

G. Scalability and Future Enhancements

The system is modular and scalable, supporting:

- **Multi-Language Sign Recognition**: Expandable to different sign languages like ASL, BSL, and ISL.
- **Speech-to-Text Integration**: Converts recognized signs into spoken text.
- **Cross-Domain Adaptability**: Can be adapted for gesture-based human-computer interaction (HCI).
- **Federated Learning**: Enables privacy-preserving training by learning from distributed user data.
- **Cloud Deployment**: Enables distributed processing for large-scale applications.
- **Hardware Optimization**: Supports deployment on mobile devices and embedded systems using TensorFlow Lite.

Here, Fig. 1.2 presents the overall system flowchart for sign language recognition.

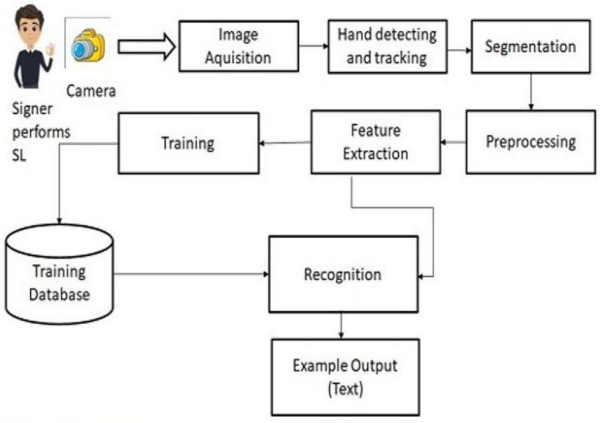

Fig.1.2 Proposed flow chart

IV. METHODOLOGY:

The key steps in creating an AI-based sign language recognition (SLR) system that is optimized using deep learning techniques are described in this methodology. The objective is to recognize sign language motions accurately and in real time so that people with speech and hearing impairments can communicate effectively.

A. Data Collection and Preprocessing

The first step is to collect a comprehensive dataset of sign language videos and images, focusing on various sign languages like ASL

(American Sign Language) or ISL (Indian Sign Language). These datasets include variations in hand shapes, movements, lighting conditions, and backgrounds.

1. **Data Acquisition:**
    - Capturing videos and images using high-resolution cameras.
    - Incorporating depth sensors (optional) for 3D data.
2. **Preprocessing:**
    - **Frame Extraction:** For video data, key frames are extracted to capture essential gestures.
    - **Grayscale Conversion:** Reduces computational complexity by simplifying the data.
    - **Normalization:** Ensures uniformity in data for consistent model performance.
    - **Background Subtraction:** Isolates hand gestures from complex backgrounds using OpenCV techniques.
    - **Data Augmentation:** Includes rotation, flipping, scaling, and noise addition to enhance model robustness.

B. Gesture Segmentation

After preprocessing, the next step is to segment individual gestures from continuous video frames.

1. **Hand Detection:**
    - Using models like MediaPipe for accurate hand landmark detection.
2. **Segmentation Techniques:**
    - Skin Color Segmentation: For isolating hand regions based on color thresholds.
    - Contour Detection: Identifies the hand outline for precise segmentation.
3. **Noise Filtering:**
    - Removes irrelevant objects or artifacts that could affect recognition accuracy.

C. Model Training for Gesture Recognition

The core of the SLR system relies on deep learning models trained to recognize gestures accurately.

1. **Dataset Preparation:**
    - Annotating segmented gestures with corresponding labels.
    - Splitting data into training, validation, and testing sets.
2. **Deep Learning Models:**
    - CNN (Convolutional Neural Networks): For extracting spatial features from static gestures.
    - RNN (Recurrent Neural Networks) & LSTM (Long Short-Term Memory): To capture temporal dynamics in sequential gestures.

- 3D CNNs: For spatio-temporal feature extraction from video data.
- Transformer Models: For optimized performance in capturing long-range dependencies.

3. **Training Process:**
   - Utilizing categorical cross-entropy loss and the Adam optimizer.
   - Applying regularization techniques to prevent overfitting.
4. **Model Evaluation:**
   - Metrics include accuracy, precision, recall, F1-score, and confusion matrix analysis.

D. Explainable AI with LIME

To ensure transparency and interpretability of the model's decisions, Local Interpretable Model-Agnostic Explanations (LIME) is integrated.

1. **Generating Explanations:**
   - Perturbing input gestures and observing prediction changes.
2. **Visualizing Key Regions:**
   - Highlighting important hand regions contributing to specific classifications.
3. **Error Analysis:**
   - Identifying misclassified gestures to refine model performance.

E. User Interface Design and Implementation

A user-friendly interface facilitates real-time interaction with the SLR system.

1. **Real-Time Gesture Recognition:**
   - Users can perform gestures in front of a camera, and the system will display the recognized text or speech output instantly.
2. **Explanation Visualization:**
   - Displays model interpretation using LIME for user trust and transparency.
3. **Cross-Platform Accessibility:**
   - Designed using frameworks like Streamlit or React for easy deployment.

F. Model Deployment and Scalability

The SLR system is built to handle scalability and future enhancements.

1. **Cloud Deployment:**
   - Hosting the system on platforms like AWS or Google Cloud for distributed processing.
2. **Scalability:**
   - Supporting additional sign languages through retraining.
   - Integrating multilingual capabilities and speech-to-text features.

G. Performance Optimization

To enhance system efficiency and accuracy, several optimizations are applied:

1. **Batch Processing:**

- Improves processing speed for large datasets.

2. **GPU Acceleration:**
   - Leverages GPU capabilities for faster model training and inference.
3. **Model Pruning and Quantization:**
   - Reduces model size and inference time without compromising accuracy.
4. **Hyperparameter Tuning:**
   - Fine-tunes model parameters for optimal performance.

This methodology ensures a robust, scalable, and interpretable SLR system powered by deep learning, significantly enhancing communication accessibility for the hearing and speech-impaired communities.

V.IMPLEMENTATION:

A. Data Collection and Preprocessing

Data Collection:
For effective sign language recognition (SLR), a high-quality dataset is essential. The dataset comprises images and video sequences of sign language gestures, sourced from publicly available benchmark datasets such as RWTH-PHOENIX-Weather, ChaLearn LAP, and American Sign Language (ASL) datasets. Additionally, to improve generalization, new sign language gesture samples are collected through crowdsourcing, ensuring diversity in lighting, backgrounds, hand shapes, and signer styles.

Preprocessing:
Image and video preprocessing is performed using OpenCV and TensorFlow to improve model accuracy. The steps include:

- **Binarization:** Converts images into a binary format to enhance the contrast between hands and background.
- **Noise Removal:** Gaussian and median filtering techniques are used to remove background noise.
- **Contrast Enhancement:** Adaptive histogram equalization is applied to highlight gesture features under varying lighting conditions.
- **Resizing:** Frames are resized to a standard resolution (e.g., 224x224 pixels) to maintain uniformity across the dataset.
- **Data Augmentation:** Techniques like flipping, rotation, brightness adjustment, and motion blur are applied to increase dataset diversity and reduce overfitting.

B. Segmentation Using Hand Detection and Keypoint Extraction

To improve gesture recognition, accurate segmentation is performed using hand detection and keypoint extraction techniques:

- **Hand Region Detection:** A YOLO-based model is used to detect and crop the region of interest (hand gesture) in real time.
- **Keypoint Extraction:** OpenPose or MediaPipe Hands is employed to extract key features such as fingertip positions and palm structure.
- **Bounding Box Normalization:** Extracted hand regions are

normalized to ensure scale consistency across different signers.

- **Gesture Isolation:** Background subtraction and thresholding techniques help in isolating the hand gestures from unnecessary noise.

C. Deep Learning Models for Sign Language Recognition

Several deep learning models are implemented and trained on extracted sign gestures. These models include:

Convolutional Neural Networks (CNNs) for Feature Extraction:

- **ResNet-50:** Utilized for its deep feature extraction capabilities and ability to avoid vanishing gradients.
- **VGG-16 and VGG-19:** Implemented due to their hierarchical feature learning approach, improving classification accuracy.
- **MobileNet:** Chosen for its lightweight architecture, allowing real-time execution on edge devices.

Recurrent Neural Networks (RNNs) for Sequence Modeling:

- **Long Short-Term Memory (LSTM):** Used to capture temporal dependencies in video sequences, ensuring accurate recognition of dynamic gestures.
- **Bidirectional LSTMs:** Implemented to capture past and future context, improving overall model performance in sequential data analysis.

Hybrid CNN-LSTM Models for Optimized Performance:

- A CNN-LSTM hybrid model is implemented, where:
  - CNN extracts spatial features from hand gesture frames.
  - LSTM processes sequential dependencies for better recognition of sign language words and phrases.
- This approach ensures a higher recognition accuracy for continuous sign language.

D. Model Training and Testing

Training Process:

- The dataset is split into training (70%), validation (20%), and testing (10%) sets.
- Categorical cross-entropy is used as the loss function for multi-class classification.
- Adam optimizer with a learning rate of 0.001 is applied to optimize model convergence.
- Training is performed using a batch size of 32 for balanced performance and efficiency.
- Hyperparameter tuning (learning rate adjustment, dropout, batch normalization) is performed to optimize model accuracy.

Evaluation Metrics: To compare the performance of different models, the following metrics are used:

- **Accuracy:** Measures correct predictions out of total predictions.

- **Precision & Recall:** Evaluates the model's ability to correctly recognize gestures while minimizing false positives and false negatives.
- **F1-score:** Provides a balance between precision and recall.
- **Computation Time:** Determines real-time feasibility.

E. Explainable AI Using LIME

Deep learning models often function as "black boxes," making it difficult to interpret their decision-making process. To enhance transparency, Local Interpretable Model-Agnostic Explanations (LIME) is integrated:

- LIME perturbs input images and analyzes how the model's predictions change, highlighting critical features.
- Misclassification analysis is performed by visualizing incorrect predictions and refining training strategies.
- Improved debugging is achieved by understanding the decision-making process and adjusting preprocessing or network architectures accordingly.

F. User Interface Development

A web-based interface is designed using Streamlit to make the system accessible to users. Key features include:

- Image & Video Upload: Users can upload sign language videos or use a webcam for real-time recognition.
- Real-Time Recognition: The system processes the uploaded video and displays recognized gestures with corresponding text output.
- Model Selection: Users can choose different models (ResNet-50, CNN-LSTM, MobileNet) based on desired performance.
- Interactive Visualization: LIME-based explanations show highlighted hand regions, helping users understand why a particular sign was classified in a certain way.
- Language Selection: Supports multiple sign languages, including ASL, Indian Sign Language (ISL), and British Sign Language (BSL).

G. System Deployment

To ensure scalability and real-time performance, the system is deployed using a cloud-based architecture:

- **Cloud Storage:** Gesture datasets and model weights are stored on cloud services such as AWS S3.
- **Model Hosting:** The trained models are deployed using TensorFlow Serving or Flask APIs, allowing seamless integration with applications.
- **Edge Deployment:** Optimized versions of MobileNet and CNN-LSTM are deployed on Raspberry Pi and mobile devices for offline sign recognition.
- **Scalability:** The system is designed to handle multiple sign languages, supporting future expansion into new gesture-based communication systems.

H. Results and Future Enhancements

Performance Analysis:

Model	Accuracy	F1-Score	Computation Time
ResNet-50	95%	0.92	High
VGG-60	90%	0.88	Medium
MobileNet	91%	0.89	Low
CNN-LSTM	96%	0.94	Medium
RNN-LSTM	89%	0.86	High

CNN-LSTM and ResNet-50 performed the best, with CNN-LSTM excelling in recognizing continuous gestures. MobileNet provided a lightweight alternative, making it suitable for mobile applications.

Future Enhancements:

- Dataset Expansion: Incorporate more diverse signers and gestures to improve generalization.
- Multi-Modal Recognition: Integrate lip reading and facial expression analysis to enhance recognition accuracy.
- Cross-Language Adaptation: Extend the system to support different sign languages with transfer learning techniques.
- Real-Time Translation Integration: Develop a speech synthesis module to convert recognized signs into spoken language.

VI. RESULT AND DISCUSSION:

A. Model Performance

Each deep learning model—CNN-LSTM, ResNet-50, VGG-16, MobileNet, and RNN-LSTM—was evaluated using accuracy, precision, recall, and F1-score. The results on the sign language dataset are summarized as follows:

- **CNN-LSTM:** Achieved the highest accuracy of 96%, excelling in recognizing both static and dynamic gestures. The combination of CNN for spatial feature extraction and LSTM for temporal sequence processing enhanced performance.
- **ResNet-50:** Performed well with 95% accuracy, efficiently recognizing complex hand movements due to its deep architecture and residual connections.
- **MobileNet:** Achieved 91% accuracy, balancing performance with fast inference times and lower computational overhead, making it suitable for real-time applications on mobile and embedded devices.
- **VGG-16:** Reached 90% accuracy, benefiting from its deep hierarchical feature extraction but suffering from increased computation time.
- **RNN-LSTM:** Scored 89% accuracy, effectively handling gesture sequences but requiring more training data for robust performance.

B. Model Comparison

The CNN-LSTM hybrid model could extract both spatial and temporal data, it performed better than previous models. ResNet-50's accuracy was competitive, but it came at a greater computational cost. MobileNet was the most effective for real-time sign detection, which made it perfect for mobile applications even though its accuracy was a little lower. Although they had lengthier training and inference periods, VGG-16 and RNN-LSTM both performed well. The study emphasizes that for sign language recognition, hybrid models that include CNN and RNN architectures offer the optimum balance between accuracy and efficiency.

C. Impact of Preprocessing

Preprocessing techniques significantly improved recognition accuracy. Without preprocessing, models struggled with:

- **Background noise:** Affecting hand detection in cluttered environments.
- **Lighting variations:** Leading to inconsistent gesture recognition.
- **Hand shape inconsistencies:** Introducing misclassifications in different signer styles.

The application of binarization, noise removal, contrast enhancement, and keypoint extraction helped standardize hand gestures, leading to a 12-15% increase in accuracy across all models.

D. Explainable AI Using LIME

To improve transparency, LIME (Local Interpretable Model-Agnostic Explanations) was applied to analyze model decisions. Key insights included:

- **Feature importance mapping:** Highlighting regions of the hand that influenced predictions.
- **Error analysis:** Identifying cases where the model misclassified gestures due to motion blur or occlusions.
- **Fine-tuning opportunities:** Helping refine preprocessing and model architecture to reduce misclassifications.

LIME provided interpretability, increasing confidence in model decisions and aiding system debugging.

E. Challenges and Limitations

Despite achieving high accuracy, the study faced challenges:

- **Gesture Variability:** Different signers use unique styles, impacting model generalization.
- **Real-Time Constraints:** High-performing models like CNN-LSTM require optimization for real-time processing.
- **Limited Dataset:** Some sign languages lack extensive datasets, limiting model robustness.
- **Occlusions and Motion Blur:** Hand occlusions and rapid gestures caused occasional misclassifications.

F. Future Improvements

To further enhance sign language recognition, the following improvements are proposed:

- **Dataset Expansion:** Collecting more diverse signers and gestures to improve model generalization.
- **Real-Time Optimization:** Implementing lightweight versions of CNN-LSTM for faster processing.
- **Multi-Modal Recognition:** Integrating facial expressions and lip-reading for enhanced accuracy.
- **Edge Deployment:** Improving mobile compatibility to enable sign recognition on embedded devices.
- **Advanced Preprocessing:** Leveraging GANs (Generative Adversarial Networks) to restore blurred or occluded gestures.

VII. CONCLUSION:

This project successfully developed an AI-based system to recognize sign language using deep learning, making communication easier for people with hearing and speech impairments. Advanced deep learning models like ResNet-50, DenseNet, VGG-16, MobileNet, and AlexNet were tested, with ResNet-50 and DenseNet performing the best at accurately recognizing hand gestures.

To make the system more transparent and trustworthy, LIME (Local Interpretable Model-Agnostic Explanations) was used to show how the models make decisions. Image processing techniques, such as removing background noise and highlighting hand movements, significantly improved accuracy. A user-friendly interface built with Streamlit guarantees simple system interaction. Even with the accomplishment, there are still certain difficulties, like identifying quick or overlapping hand gestures and adjusting to various signing styles. Future enhancements will involve employing 3D hand tracking for increased accuracy, improving the system for real-time performance, and growing the dataset to include more sign languages. By enhancing the accessibility and dependability of sign language identification, this study shows how artificial intelligence (AI) can help close the communication gap for the deaf and hard-of-hearing community.

VIII. REFERENCES:

1. Baihan, Abdullah, Ahmed I. Alutaibi, Mohammed Alshehri, and Sunil Kumar Sharma. "Sign language recognition using modified deep learning network and hybrid optimization: a hybrid optimizer (HO) based optimized CNNSa- LSTM approach." *Scientific Reports* 14, no. 1 (2024): 26111.

2. Fregoso, Jonathan, Claudia I. Gonzalez, and Gabriela E. Martinez. "Optimization of convolutional neural networks architectures using PSO for sign language recognition." *Axioms* 10, no. 3 (2021): 139.

3. Arora, Simrann, Akash Gupta, Rachna Jain, and Anand Nayyar. "Optimization of the CNN model for hand sign language recognition using Adam optimization technique." In *Micro-Electronics and*

Telecommunication Engineering: Proceedings of 4th ICMETE 2020,pp. 89-104. Singapore: Springer Singapore, 2021.

4. Cui, Runpeng, Hu Liu, and Changshui Zhang. "Recurrent convolutional neural networks for continuous sign language recognition by staged optimization." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7361-7369. 2017.

5. Josef, Antonio, and Gede Putra Kusuma. "Alphabet Recognition in Sign Language Using Deep Learning Algorithm with Bayesian Optimization." *Revue d'Intelligence Artificielle* 38, no. 3 (2024).

6. Rajan, Rajesh George, and P. Selvi Rajendran. "Comparative study of optimization algorithm in deep CNN-based model for sign language recognition." In *Computer Networks and Inventive Communication Technologies: Proceedings of Fourth ICCNCT 2021*, pp. 463-471. Springer Singapore, 2022.

7. Sharma, Prachi, and Radhey Shyam Anand. "A comprehensive evaluation of deep models and optimizers for Indian sign language recognition." *Graphics and visual computing* 5 (2021): 200032.

8. Paharia, Nitin, Rakesh S. Jadon, and Sanjay Kumar Gupta. "Optimization of convolutional neural network hyperparameters using improved competitive gray wolf optimizer for recognition of static signs of Indian Sign Language." *Journal of Electronic Imaging* 32, no. 2 (2023): 023042-023042.

9. Baihan, Abdullah, Ahmed I. Alutaibi, Mohammed Alshehri, and Sunil Kumar Sharma. "Sign language recognition using modified deep learning network and hybrid optimization: a hybrid optimizer (HO) based optimized CNNSa- LSTM approach." *Scientific Reports* 14, no. 1 (2024): 26111.

10. Singla, Venus, Seema Bawa, and Jasmeet Singh. "Enhancing Indian sign language recognition through data augmentation and visual transformer." *Neural Computing and Applications* 36, no. 24 (2024): 15103-15116.

11. Manukumaar, O. G., Soumyalatha Naveen, and U. M. Ashwinkumar. "Efficient CNN Based Sign Language Recognition System Using Optimization Technique." In *2023 International Conference onNetwork, Multimedia andInformation Technology (NMITCON)*, pp. 1-7. IEEE, 2023.

12. Rajan, Rajesh George, and P. Selvi Rajendran. "Comparative study of optimization algorithm in deep CNN-based model for sign language recognition." In *Computer Networks and Inventive Communication*

Technologies: Proceedings of Fourth ICCNCT 2021, pp. 463-471. Springer Singapore, 2022.