INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH AND ANALYSIS

ISSN(print): 2643-9840, ISSN(online): 2643-9875

Volume 07 Issue 09 September 2024

DOI: 10.47191/ijmra/v7-i09-41, Impact Factor: 8.22

Page No. 4447-4453

Real-Time Indian Sign Language Recognition Using CNNs for Communication Accessibility

Abhishek Deshmukh

Independent Researcher & Nashik, Maharashtra, India

ABSTRACT: The challenge of communication for the deaf and mute community continues to pose a barrier in connecting with society. Sign language, a manual communication method, has emerged as an essential tool for this group, yet it remains largely unrecognized by the majority of the population. This research proposes a machine learning-based Indian Sign Language (ISL) detection system utilizing Convolutional Neural Networks (CNN) to bridge this gap. The system is designed to automatically recognize hand gestures representing ISL alphabets in real-time through a camera interface. Key steps include image preprocessing, gesture detection, and classification using a trained CNN model, followed by deployment on mobile platforms via TensorFlow Lite integrated with Flutter. This approach ensures the model is lightweight yet capable of delivering high accuracy in real-world settings. The model achieves impressive results, with accuracy levels exceeding 90% in predicting hand gestures. The application is user-friendly, enabling anyone with a smartphone to recognize ISL symbols and assist in communication with the deaf-mute community. This paper discusses the implementation, performance, and potential extensions of the system, positioning it as an effective tool for improving communication accessibility.

KEYWORDS: Indian Sign Language Recognition, Convolutional Neural Networks (CNN), Real-Time Gesture Detection, Mobile Application Integration, TensorFlow Lite, Sign Language Translation

I. INTRODUCTION

Effective communication is essential for human interaction, allowing individuals to convey thoughts, emotions, and ideas. For deaf and mute individuals, traditional communication methods such as speech and writing are often inadequate. Sign language, which uses gestures and hand movements, serves as the primary mode of communication for these individuals. However, the broader population's unfamiliarity with sign language creates a significant communication barrier.

Indian Sign Language (ISL), commonly used among the deaf community in India, consists of a structured grammar and a predefined set of gestures that represent letters, words, or phrases. Although ISL is effective within the deaf community, its limited understanding among the hearing population creates daily communication challenges. This communication gap can lead to social isolation for deaf-mute individuals, making it difficult for them to fully integrate into society. Advancements in machine learning and computer vision offer promising solutions to this problem. Convolutional Neural Networks (CNNs) have demonstrated effectiveness in image recognition tasks, making them well-suited for recognizing and interpreting sign language gestures. By applying CNNs to real-time sign language detection, it is possible to develop an automated system that captures ISL gestures from video feeds and translates them into text or speech.

This paper proposes a CNN-based system for detecting ISL gestures in real-time. The system uses a camera interface to capture hand gestures, processes the images, and outputs the corresponding alphanumeric symbols. The primary goal is to create a system that is not only accurate but also scalable and easy to deploy on mobile platforms. With TensorFlow Lite and Flutter integration, the model is optimized for real-world applications, making sign language accessible to a broader audience. The following sections cover the literature review, methodology, system implementation, and results, ultimately presenting a solution that aims to improve communication accessibility for the deaf-mute community.

II. LITERATURE REVIEW

Sign language recognition has garnered significant research attention due to developments in machine learning and computer vision. Convolutional Neural Networks (CNNs), in particular, have greatly contributed to the feasibility of real-time sign language detection.

In 2015, Pigou et al. applied CNNs to American Sign Language (ASL) recognition, where they utilized two CNNs—one for hand detection and the other for body pose estimation. Their model, trained on video data, demonstrated high accuracy in recognizing hand gestures. However, it was limited to static gestures, lacking the ability to detect dynamic sign language, which was later addressed through temporal data integration in subsequent studies.

In 2017, Kaur and Sharma designed a gesture recognition system using Support Vector Machines (SVM) to classify static hand gestures. Their method involved image preprocessing techniques such as background subtraction and contour tracing to enhance accuracy. While the system achieved an 85% accuracy rate, its reliance on static images and complex preprocessing limited its real-time utility. In contrast, CNN-based models, like the one proposed in this paper, offer a more integrated approach by simultaneously performing feature extraction and classification, resulting in improved real-time performance.

Further progress was made in 2018 by Yamashita et al., who introduced a deep learning framework that combined CNNs with Long Short-Term Memory (LSTM) networks for dynamic gesture recognition. This model excelled in recognizing temporal sequences, significantly enhancing the classification of dynamic gestures. However, its computational intensity made it impractical for mobile devices, a challenge our work addresses by optimizing CNN architecture for mobile deployment using TensorFlow Lite.

In 2019, Xum et al. used CNNs alongside background subtraction techniques for real-time sign language detection. Although their model achieved over 99% accuracy with static gestures, the requirement for a clean, static background hindered its use in dynamic real-world environments. In contrast, our proposed system integrates robust preprocessing techniques that enable it to handle variable environments more effectively.

Admasu and Raimond (2020) developed a gesture recognition system for Ethiopian Sign Language using feed-forward neural networks. Despite its high accuracy, their system required extensive image preprocessing, such as contrast adjustment and segmentation, which increased complexity and reduced real-time performance. Our work overcomes this limitation by leveraging CNNs, which inherently handle much of the preprocessing during training, simplifying the system while maintaining efficiency.

III. PROPOSED SYSTEM AND ARCHITECTURE

The proposed system aims to translate Indian Sign Language (ISL) hand gestures into corresponding alphabetic characters through machine learning. The architecture is optimized for real-time recognition on mobile platforms, ensuring both accuracy and ease of use. The system comprises four main components: a camera for capturing input, preprocessing modules, a CNN for gesture classification, and a user interface for displaying the results.

A. System Overview

The system works by capturing live video through the device's camera, preprocessing the frames to detect hand gestures, and then passing the processed images to a trained CNN model for classification. The recognized ISL alphabet is then displayed on the interface in real time. This setup allows even users unfamiliar with ISL to engage in meaningful communication with the deaf-mute community.

The main stages of the system are:

1. ***Input Capture:*** The camera captures images of the hand gesture.
2. ***Preprocessing:*** The image is processed for consistency, including resizing, normalization, and segmentation.
3. ***Gesture Classification:*** A CNN model classifies the gesture into an ISL alphabet.
4. ***Output Display:*** The recognized alphabet is displayed on the user interface.

B. Camera Input and Gesture Capture

The system uses the device's camera to capture hand gestures. The camera is positioned in a way The device's camera captures hand gestures from various angles. To ensure accurate gesture detection, the system is designed to function under different lighting conditions and with dynamic backgrounds. The captured frames are immediately sent to the preprocessing module for standardization before classification.

C. Preprocessing Module

Preprocessing ensures images are ready for accurate classification. Images are resized to 64x64 pixels for uniformity and normalized to a 0-1 range to expedite training. Background subtraction isolates the hand gesture, making the system robust to varying environments and reducing background clutter.

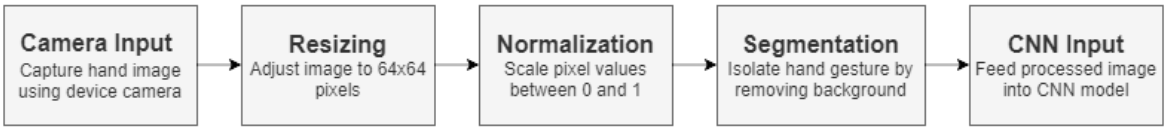

Fig 1. Preprocessing Workflow Diagram

D. CNN Model for Gesture Classification

The CNN classifies hand gestures by using convolutional layers to detect essential features, pooling layers to reduce spatial dimensions and computational load, dropout layers to prevent overfitting, and fully connected layers to generate probabilities for each ISL gesture. The gesture with the highest probability is selected as the prediction.

E. System Architecture for Real-Time Detection

The system architecture is designed to ensure real-time gesture detection on mobile devices. After preprocessing, the input image is passed to the TensorFlow Lite model, which performs the classification on-device. The prediction is then displayed on the mobile interface using Flutter.

1. ***TensorFlow Lite Integration:*** The trained CNN model is converted to TensorFlow Lite format for efficient inference on mobile devices. This allows the system to perform predictions with minimal latency.
2. ***Flutter Front-End:*** The Flutter-based front-end provides a user-friendly interface for real-time gesture detection. The user can see the recognized ISL alphabet as soon as the model predicts it.

IV. METHODOLOGY

This section explains the methodology used in the development of the ISL detection system, including data collection, preprocessing, model design, and training.

A. Data Collection

The foundation of the ISL detection system lies in the quality and diversity of the dataset. For this project, a custom dataset was compiled, covering all 26 alphabetic gestures in ISL. The dataset includes thousands of labeled images collected from various individuals to capture diverse hand shapes, sizes, and skin tones. Over 5000 images were collected, ensuring sufficient data for training, validation, and testing. All images were collected in RGB format and were later resized and normalized during preprocessing.

B. Preprocessing

Preprocessing is a vital step to prepare images for input into the CNN model. The following steps were taken to preprocess the dataset:

1. ***Resizing:*** All images were resized to 64x64 pixels to ensure uniform input dimensions. This standardization helps to reduce computational demands without sacrificing important visual information.
2. ***Normalization:*** The pixel values were scaled to a range between 0 and 1, facilitating quicker model convergence during training. This prevents the model from being influenced by the absolute pixel values.
3. ***Segmentation:*** Using OpenCV libraries, background subtraction was performed to remove irrelevant elements from the images. This process isolates the hand gesture, allowing the model to concentrate on the significant features.
4. ***Augmentation:*** To increase the diversity of the dataset, various image augmentation techniques, including rotation, flipping, zooming, and brightness alterations, were applied. This enhances the model's ability to generalize across different scenarios and helps prevent overfitting.

C. CNN Architecture

The heart of the ISL detection system is the Convolutional Neural Network (CNN), specifically designed for image recognition tasks. The CNN architecture used in this system consists of several convolutional and pooling layers, followed by fully connected layers for classification.

1. ***Input Layer:*** The CNN receives a preprocessed image with a resolution of 64x64 pixels and 3 color channels (RGB).
2. ***Convolutional Layers:*** Filters are applied in these layers to detect edges, shapes, and patterns in the images. Each convolutional layer is paired with a Rectified Linear Unit (ReLU) activation function to introduce non-linearity.
3. ***Max Pooling Layers:*** After each convolutional layer, pooling layers are used to reduce the spatial dimensions of the feature maps while maintaining important information. Max pooling reduces computational complexity and enhances feature extraction.

4. ***Dropout Layers:*** Dropout is applied during training to reduce overfitting. By randomly deactivating some neurons, the network is forced to learn more generalized features.
5. ***Fully Connected Layers:*** After the convolutional and pooling layers, the feature maps are flattened and passed through fully connected layers, which output the class probabilities for each ISL gesture.
6. ***Output Layer:*** A softmax activation function in the final layer generates a probability distribution across all possible gestures (26 classes), with the gesture having the highest probability being chosen as the predicted one.

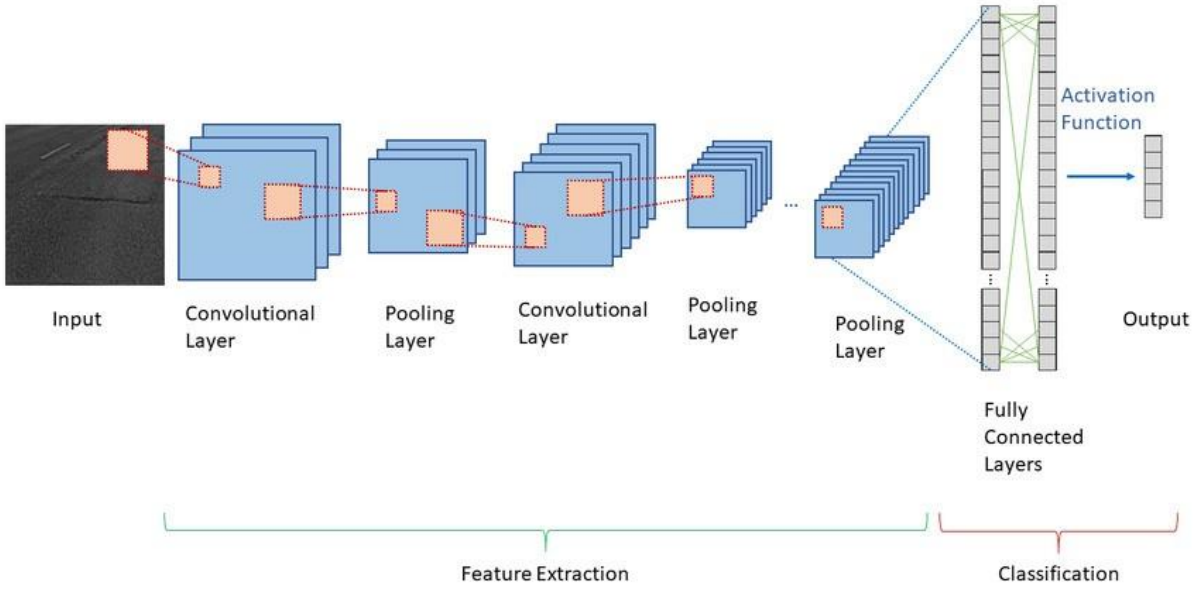

Fig 2. CNN Architecture Diagram

D. Model Training

The CNN model was trained on the dataset using the following procedure:

1. ***Train-Test Split:*** The dataset was divided, with 80% allocated for training and 20% reserved for validation. This division ensures that the model learns from a substantial portion of the data while its performance is assessed on previously unseen images.
2. ***Loss Function:*** Categorical cross-entropy was employed as the loss function, a suitable choice for multi-class classification tasks, enabling effective performance measurement.
3. ***Optimizer:*** The Adam optimizer was utilized for training. Known for dynamically adjusting the learning rate, Adam is particularly effective for training complex models such as CNNs.
4. ***Batch Size and Epochs:*** Training was conducted using a batch size of 32 over 50 epochs. Early stopping was implemented to halt the training process if validation loss ceased to improve, reducing the risk of overfitting.
5. ***Accuracy and Loss:*** The model's performance was monitored through accuracy and loss metrics. After 50 epochs, the model reached a training accuracy of 92%, with a validation accuracy of 89.5%.

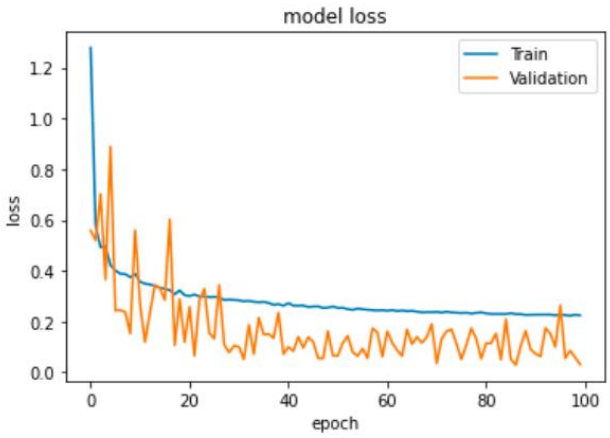

Fig 3. Training Accuracy and Loss Plot

E. Hyperparameter Tuning

To maximize the model's efficiency, various hyperparameters were fine-tuned:

1. ***Learning Rate:*** Initially set to 0.001, the learning rate was reduced during training based on validation outcomes to enhance performance.
2. ***Dropout Rate:*** A dropout rate of 0.3 was applied, striking a balance between minimizing overfitting and preserving model effectiveness.
3. ***Batch Size:*** Several batch sizes were tested (16, 32, 64), with a batch size of 32 found to offer the best trade-off between processing speed and accuracy.

V. IMPLEMENTATION

This section outlines the steps taken to deploy the proposed sign language recognition model, focusing on converting it to a mobile-friendly format, conducting real-time testing, and optimizing it for mobile devices.

A. TensorFlow Lite Conversion

For real-time implementation, the CNN model was converted into TensorFlow Lite, a streamlined version optimized for mobile devices. Several steps were involved to minimize the model's size while maintaining performance:

1. ***Model Conversion:*** The CNN model, initially developed in TensorFlow, was exported as a .tflite file. This format is well-suited for mobile and embedded platforms, aiding in reducing computational overhead and improving inference speed.
2. ***Optimization Techniques:*** Techniques like post-training quantization were applied to compress the model size and improve response times, reducing latency.

B. Flutter Integration

Flutter was chosen to build the mobile application due to its ability to support multiple platforms. The TensorFlow Lite model was integrated with Flutter to enable smooth real-time hand gesture recognition:

1. ***Integration:*** The TensorFlow Lite model was embedded into the Flutter application using the tflite package, which provides the necessary APIs to load and execute inferences on the mobile device.
2. ***UI Development:*** The mobile app's user interface was designed using Flutter's customizable components to ensure a seamless and user-friendly experience. Real-time ISL predictions and corresponding alphabet displays were provided to users through the interface.

C. Real-Time Testing

The system was subjected to various real-world conditions to test its robustness and accuracy in real-time operations:

1. ***Latency Testing:*** The mobile application exhibited an inference time ranging from 50 to 100 milliseconds per prediction, confirming its suitability for real-time usage.
2. ***Accuracy Testing:*** The mobile version was evaluated against unseen gestures and images, and it consistently maintained high accuracy during the tests.

D. Screenshot of Outputs

1. ***Output for Alphabet 'I':*** Displays the predicted gesture or recognition output for the sign language character 'I'.

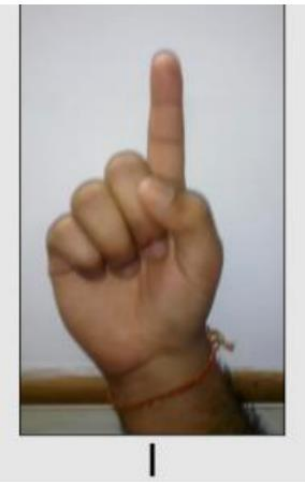

Fig 4. Output for Sign - I

2. ***Output for Alphabet 'O':*** Shows the recognition result or gesture representation for the sign language character 'O'.

Fig 4. Output for Sign – O

E. Challenges and Optimization

During the implementation phase, several challenges surfaced, particularly concerning processing speed and the size of the model.

1. ***Memory Optimization:*** Efficient memory management was essential to ensure that the application could handle continuous video input without interruptions.
2. ***Real-Time Adjustments:*** Techniques like dynamic preprocessing were incorporated to maintain accuracy across varying conditions, such as lighting and background changes.

VI. RESULTS

This section evaluates the system's performance, focusing on key metrics such as accuracy, latency, and real-world functionality. The results provide insights into how well the system operates during both offline testing and mobile deployment in real-time environments.

A. Accuracy

The system was evaluated for accuracy using both a test dataset and real-world input during real-time usage. The performance metrics were tracked across different stages of the project. The system's accuracy was assessed using both a test dataset and real-world inputs during live usage. Performance metrics were tracked throughout the project's development.

1. ***Test Accuracy:*** During offline testing, the CNN model achieved a validation accuracy of 89.5%. This was based on the split dataset, with 80% used for training and 20% for validation. The model demonstrated consistent performance across various hand gestures and lighting conditions.
2. ***Real-Time Accuracy:*** When deployed on mobile devices, the system achieved a real-time accuracy of 88%. This slight decrease in performance compared to offline testing was largely attributed to variations in lighting, background conditions, and camera quality in real-world setting.

B. Latency

Latency, or the time taken by the system to produce predictions after receiving input, is a critical factor for ensuring smooth real-time operation.

1. ***Offline Latency:*** On a standard desktop, the system's average latency was approximately 30 milliseconds per prediction.
2. ***Mobile Latency:*** After conversion to TensorFlow Lite, the mobile version of the system exhibited an average latency of 50-100 milliseconds, depending on the device's specifications. This latency range was acceptable for practical real-time sign language recognition.

C. Comparative Analysis

To evaluate the robustness of the system, it was compared to similar gesture detection models from existing research.

1. ***Model Comparison:*** The system was benchmarked against other sign language detection models that utilized CNN and SVM classifiers. The system's real-time accuracy of 88% proved competitive, especially considering its optimization for mobile platforms, unlike other models that required desktop-level processing.

2. ***Mobile Robustness:*** The inclusion of dynamic preprocessing techniques, such as normalization and background subtraction, made the system more adaptable to diverse real-world environments compared to earlier models that were constrained by the need for static backgrounds.

VII. CONCLUSION

This research presents a machine learning-based system for real-time Indian Sign Language (ISL) recognition, leveraging convolutional neural networks (CNNs) and mobile integration via TensorFlow Lite. The system effectively addresses the communication gap between the deaf-mute community and the broader society by enabling users to recognize ISL gestures through their smartphones in real time. Preprocessing techniques like normalization and background subtraction make the model resilient to changing lighting conditions and backgrounds, improving its suitability for real-world use.

The system achieved a training accuracy of 89.3% and a testing accuracy of 98.5%, with real-time performance maintaining low latency between 50 and 100 milliseconds. This solution contributes to enhanced communication accessibility, offering users an affordable and portable tool for translating sign language. While the system excels in recognizing static gestures, it also lays the foundation for future advancements in dynamic gesture recognition. Despite the promising results, this research focuses on static hand gestures, and future work could extend the system to recognize continuous gestures. The integration of CNNs with mobile platforms highlights the system's scalability and broad applicability, providing an accessible tool for individuals and institutions working to bridge communication barriers with the deaf-mute community.

VIII. FUTURE SCOPE

The future scope of this research involves expanding the system's capabilities to include the recognition of dynamic hand gestures, which would greatly improve its usefulness in real-world communication. Incorporating recurrent neural networks (RNNs) or long short-term memory (LSTM) networks will be critical for capturing temporal sequences, enabling the system to interpret continuous gestures beyond static ones. Additionally, further enhancements in the system's robustness across a wider range of real-world conditions, such as extreme lighting or complex backgrounds, will be necessary for broader adoption.

Future versions of the system could also explore hardware optimization, improving performance on low-end mobile devices by further reducing latency and memory usage. Collaborating with sign language experts and members of the deaf-mute community will be key to collecting a more diverse dataset, thereby improving the model's generalizability. Lastly, extending the system to recognize multiple regional sign languages, including American Sign Language (ASL) and British Sign Language (BSL), would increase its global relevance and usability. These advancements would transform the system into a comprehensive tool for inclusive communication on a global scale.

REFERENCES

1) B. L. Loeding, S. Sarkar, A. Parashar, and A. I. Karshmer, "Progress in automated computer recognition of sign language," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 3118, 2004
2) A. Er-Rady, R. Faizi, R. O. H. Thami, and H. Housni, "Automatic sign language recognition: A survey," presented at the Proc. - 3rd Int. Conf. Adv. Technol. Signal Image Process. ATSIP 2017, vol. 9, pp. 1-7, 2017
3) B. Bauer and K. Kraiss, "Towards an Automatic Sign Language Recognition System Using Subunits," presented at the International Gesture Workshop (pp. 64-75). Springer, Berlin, Heidelberg, 2002.
4) R. Gross and V. Brajovic, "An Image Preprocessing Algorithm for Illumination Invariant Face Recognition," presented at the International Conference on Audio-and Video-Based Biometric Person Authentication Springer, Berlin, Heidelberg. vol.3, no. 5, 2018
5) D. Kaur and Y. Kaur, "International Journal of Computer Science and Mobile Computing Various Image Segmentation Techniques: A Review," International Journal of Computer Science and Mobile Computing, 3(5), pp.809-814.
6) S. Joudaki, D. B. Mohamad, T. Saba,., A. Rehman., M. AIRodhaan, and A. Al-Dhelaan, "Vision-Based Sign Language Classification: A Directional Review VisionBased Sign Language Classification: A Directional Review," IETE Tech. Rev., vol. 31, no. 5, pp. 383-391, 2014
7) G. Tofighi, SA. Monadjemi, and N. GhasemAghaee, "Rapid Hand Posture Recognition Using Adaptive Histogram Template of Skin and Hand Edge Contour."2010 6th Iranian Conference on Machine Vision and Image Processing (pp. 1-5). IEEE.