DOI - 10.68648/ijte-q-2026-49-03-014

ResNet50 Based Sign-to-Text-to-Speech Translation

Sheetal A. Takale
Information Technology Department
Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology
Baramati, Maharashtra
✉ sheetaltakale@gmail.com

Vaishnavi D. Jankar
Information Technology Department
Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology
Baramati, Maharashtra
✉ vaishnavijankar28@gmail.com

ABSTRACT

Today, effective communication is important in our connected world. However, for millions of deaf and mute people, daily communication is difficult. Sign language allows deaf and mute people to communicate with others. However, many people with normal hearing do not understand sign language. Existing automated sign language recognition systems often have low accuracy and perform poorly in different background conditions. This limits their use in real-world applications. This project aims to solve these problems. It develops a system that translates sign language into text and speech in real time. At its core, the system utilizes a custom CNN model and a pre-trained Convolutional Neural Network - ResNet50 to precisely detect and interpret manual gestures. Once the sign is recognized, it will be converted into text and then into speech. This research compares performance of custom CNN model and ResNet50 mode. The custom CNN model achieved accuracy of 88.2%. The ResNet50 model achieved a classification accuracy of 94%, demonstrating its effectiveness in sign language gesture recognition.

KEYWORDS *: Sign Language Recognition, ResNet50, CNN, Gesture Recognition, Real-time Communication, Speech Conversion.*

INTRODUCTION

As per the World Health Organization (WHO) records, about 466 million people worldwide have hearing loss. This represents more than 5% of the global population. Hearing loss is the third most common physical disability in the world. Among those affected, about 34 million are children. Conventional communication between deaf and hearing individuals largely relies on human interpreters. However, availability of human interpreter is often limited. Also services of human interpreter can be costly and inconsistent in real-time environments. Around 72 million deaf individuals use sign language as their primary mode of communication, relying on more than 300 recognized sign languages worldwide. Sign languages used worldwide are listed in Table 1. However, less than 0.1% of people with normal hearing ability know sign language. This creates a major communication gap in the society. [1]. This results in an immediate need of automated sign language recognition and translation systems.

In the present research on sign language recognition and translation systems many of them use gloves with sensors. Issues to be addressed with such system are: cost effectiveness, discomfort resulting due to need of wearing it for long periods, and correctness of results [2]. This restricts their usability in everyday communication scenarios. A major This work is open access and licensed under the Creative Commons CC BY 4.0 License. Volume 12 (2022), Issue 6 limitation of many glove-based systems is need of static hand gestures. Their performance on dynamic gestures is limited. [3]. Existing vision-based glove-free approaches require controlled environments, including plain backgrounds, stable illumination, and fixed camera positioning, to achieve acceptable performance and improved accuracy [4]. Furthermore, several traditional and lightweight deep learning models exhibit high inference latency and limited classification accuracy, which constrains their applicability in real-time assistive communication systems [5].

The proposed system addresses these challenges using

a standard camera and the ResNet50 deep learning model unlike previous systems. It does not need special hardware, gloves, or a controlled environment [6]. The proposed solution is designed with simplicity in mind. User just has to perform natural sign language gestures. The system automatically recognizes the gestures. Deep learning model converts the recognized signs into text and speech. ResNet50 is a deep CNN model with residual learning. It is pretrained on ImageNet weights for transfer learning. It assures high accuracy and is well suited for sign language recognition [7]. It suitable for real-world applications becuase it is capable of handling variations in backgrounds and lighting conditions [8]. It assures accuracy for both static and dynamic hand signs in real-time. This ensures faster and more accurate results [9]. This project aims to improve the performance and practicality of sign language recognition in order the communication between deaf and hearing people. [10].

Table 1. Sign Languages by Region Summary

Region	Number of sign languages	Notable Examples
North America	10+	ASL, LSM, Maritime Sign Language
South America	15+	Libras, Argentinian SL, Colombian SL
Europe	30+	BSL, DGS, LSF, SSL
Asia	50+	JSL, CSL, ISL, KSL
Middle East	20+	Arabic SL, Israeli SL, Iranian SL
Africa	40+	SASL, Nigerian SL, Kenyan SL
Oceania	5+	Auslan, NZSL
Indigenous / Isolated	130+	Various village sign languages

LITERATURE REVIEW

Computer vision and deep learning has a major application for the problem of Sign language recognition (SLR) to develop a good communication between deaf and hearing people. Researchers have applied various technologies and methods for sign language recognition and translation including sensor-based, vision-based, and deep learning methods.

Preliminary approaches to Sign language recognition (SLR) use sensor-based systems to capture hand movements. Wearable devices like smart gloves equipped with flex sensors and accelerometers were used to detect finger positions and orientations. These approaches achieved an accuracy up to 96% for 20 letters of the alphabet.

Some systems use Arduino boards, flex sensors, and GSM modules to convert hand gestures into text and speech. Some use cloud-based learning to predict gestures. However, these systems require special hardware. This increases cost and reduces portability. They are also difficult to scale. As a result, vision-based methods have become more popular because they do not require specialized equipment.

Thakar et al. [10] proposed a model utilizing transfer learning with VGG16, achieving 98.7% accuracy. Their approach demonstrated high accuracy. However, it relied heavily on pre-trained models. This resulted in limitation on flexibility for custom datasets and new sign variations.

Duraisamy et al. [11] proposed a system where gesture recognition is done using convolution neural network, textual translation is done through NLP, and for speech synthesis neural TTS is used. Proposed approach achieved an accuracy of 97.6%. However, the use of multiple sequential deep learning components has resulted in high computation, slow model execution, and challenges in implementation on low resource devices.

For ASL finger spelling translation Munde et al. [12] developed a Deep Learning CNN model. Work carried out in this research particularly focuses on hand orientation and posture. Results obtained in this research show strong prediction performance. However, limitation of the system was limited performance to isolated static alphabet gestures. Proposed approach was unsuitable for continuous or dynamic sign sequences.

YOLOv5 for real-time sign recognition was proposed by Fathima et al. [13]. Proposed approach achieved Mean Average Precision (mAP) between 92% and 99% for 15 gesture classes.

Parashar et al. [4] compared performance of CNN model with ResNet50. This research reported that CNN models often miss minute gesture details. Where as Deep CNN ResNet50 can capture the minute details, enhancing recognition accuracy up to 88.7%. However, deeper networks increase computational load, and model struggled with complex hand gestures and background noise, making real-time deployment challenging on low-end devices.

Likewise, Barbhuiya et al. [7] used AlexNet and VGG16 architectures for Sign Language Recognition. This work succeeded with an accuracy of 99.82%. However, major concern was their experiments relied on relatively small datasets.

Stergiopoulou and Papamarkos [14] developed custon CNN model and achieved effective performance with 98% recognition accuracy. However, the approach proposed suffered due to requirement of high-performance hardware for high computational complexity.

Approach proposed by Haidar and Reefat [2] used sensor based gloves. It enabled highly accurate fine-grained gesture capture. However, the proposed approach suffered due to requirement of wearable hardware. It significantly reduced user comfort, portability, and scalability.

Approach proposed by Yan et al. [3] implemented Kinect-based depth sensing for dynamic sign recognition. It improved motion tracking accuracy. However, it suffered due to increase in overall system cost and hardware dependency.

In contrast, Singh et al. [15] utilized depth cameras to enhance gesture segmentation and to significantly improve the accuracy. Their approach enhanced gesture tracking but required expensive depth cameras.

Sequence modelling techniques is also used for dynamic gesture and temporal sign recognition. Abraham et al. [16] explored LSTM-based temporal modelling with glove sensors. It achieved a reliable dynamic gesture recognition but it requires a large sequential dataset, long training time, and significant computational resources. Similarly, Murakami and Taguchi [17] found that RNN-based models are computationally expensive and often struggle with unseen user patterns and signer variability. Xu et al. [18] implemented the S2VT sequence-to-sequence framework. However, the model suffered from high memory consumption, limiting its real-time applicability.

Garg et al. [19] proposed a deep learning-based sign language translator using advanced neural network architectures. Their system effectively converted sign gestures into text. However, their model was primarily focused on isolated gestures. It did not address dynamic gesture sequences or context-based sentence formation. This limits its usability for real-world conversations.

Survey of hand gesture recognition approaches proposed by Nandwana et al. [6] reported approaches which have accuracies exceeding 95%, however have issues regarding real-time deployment constraints, hardware optimization, or practical usability challenges. Mohamed et al. [1] identified that many existing SLR systems perform well on trained users howevre fail to generalize to new signers, dialectal variations, and uncontrolled environments. Table 2 presents a comparative analysis of recent deep learning-based sign language recognition studies published during 2025–2026. The comparison includes the deep learning model employed, dataset used, performance metrics such as accuracy, precision, recall, and F1-score (where reported), major findings, and identified limitations.

METHODOLOGY

Dataset

A dataset of American Sign Language (ASL) downloaded from Kaggle has image samples of of hand gestures with wide variety which are commonly used in ASL communication. The dataset has hand gestures for English alphabets, numbers, and additional commonly used symbolic gestures. It is organized into 43 distinct class folders, with each folder containing a unique gesture category.

The training dataset has images with 224 × 224 pixels resolution with a total of 105,600. These images have compatibility with deep learning architectures such as ResNet50. As listed in table 3, the gesture classes include 26 classes for alphabet signs (A–Z), 10 classes for numbers or numerical signs (0–9), and several additional gesture categories which are used in practical sign language communication. Each class contains

approximately 2,400 images, providing a balanced distribution across categories and reducing class imbalance during training.

However, some of the images in the dataset which are at a lower resolution of 50 × 50 pixels, are resized during pre-processing to preserve uniformity in feature extraction.

The diversity and balanced class representation of this dataset make it important benchmark for training robust ASL gesture classification models, accurate sign interpretation. The dataset is available at Kaggle - American Sign Language .

Table 2. Overview of the ASL Dataset

Category	Classes	Images per Class
Alphabets (A-Z)	26	2,400
Numbers (0-9)	10	2,400
English Words	7	2,400
Total	43	105,600

Data Pre-processing and Image Processing

Data pre-processing is a key step for improving the overall performance and accurate prediction on unseen data for deep learning models. In the proposed system, the first pre-processing step is all input images are first resized to a fixed resolution of 224 × 224 pixels. Figure 8 shows the images before and after resizing. It ensures dimensional consistency and compatibility with the ResNet50 architecture. Using fixed input size enables efficient batch processing and uniform feature extraction across the entire dataset.

Next step in pre-processing is normalization, the pixel values are divided by 255. This normalizes the values from 0–255 to 0–1. This normalization step help stabilize the gradient updates. It also accelerates convergence during training, and improves the network's ability to learn discriminative visual features.

Third step of pre-processing is data augmentation. This technique is used to improve model robustness and reduce the risk of overfitting. The data augmentation techniques used in this research include rotation, random horizontal flip. These augmentation techniques increased the size of dataset and diversity of the training data. It also improves model performance even under different hand positions, lighting conditions, backgrounds, and poses.

After pre-processing, for training and evaluation of the model, the dataset is divided into 80% training and 20% validation sets.

The proposed system architecture, shown in Figure 2, is designed as a complete real-time sign language recognition (SLR) and translation to text and speech framework. The process begins with the camera input module, where real-time video frames containing hand gestures are captured using a standard camera.

The captured frames are further pre-processed. Pre-processing involves operations such as resizing of images, color space conversion, noise filtering using Gaussian blur and thresholding, contour extraction, and normalization. These steps first isolate the hand region from the background, then reduce noise and effects of lighting variations.



Fig 1. Comparison of images before and after resizing (224x224)

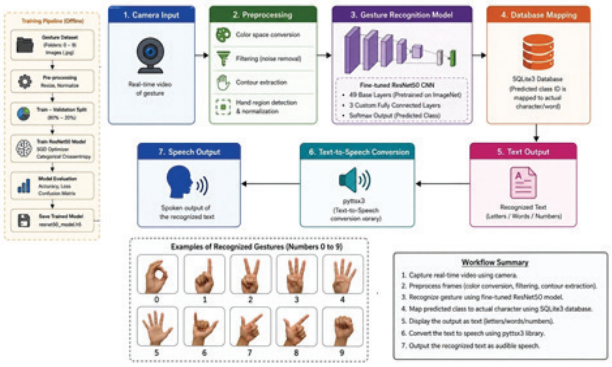


Fig 2. System Architecture

The pre-processed gesture images are given as input to the fine-tuned ResNet50 model with residual connections. It uses 49 pre-trained layers and 3 custom

classification layers to extract features and classify hand gestures. The residual connections help the model train faster and improve recognition accuracy.

The predicted gesture is matched with its corresponding letter, word, or number using an SQLite3 database. The database acts as a lookup table and converts the prediction into corresponding text. This design makes it easy to add new gestures in the future.

The mapped result is the text of the recognized letter, word, or number. The text is then converted into speech using the pyttsx3 text-to-speech (TTS) library. This allows users to hear the recognized sign and improves accessibility.

Hyper parameter such as the learning rate, batch size, number of epochs, and optimizer are adjusted to improve model performance. Dropout layers are added to reduce overfitting and improve the model's performance on new gesture images.

Proposed approach combines the steps of video acquisition, image pre-processing, ResNet50-based deep feature extraction and SLR classification, database-assisted semantic mapping, and speech synthesis. This proposed architecture provides an efficient and practical assistive communication system capable of functioning in diverse real-world environments.

Algorithm 1 Real-Time Sign Language Recognition and Speech Conversion

```
Initialize:
  Initialize webcam: cap = cv2.VideoCapture(0)
  Load model: model = load_model('resnet50.h5')
Define Preprocessing Function:
  def preprocess(frame):
    frame = cv2.resize(frame, (224, 224))
    frame = np.expand_dims(frame, axis=0)
    frame = frame / 255.0
    return frame
Start Video Capture Loop:
  while True:
    ret, frame = cap.read()
    if not ret: break
    roi = preprocess(frame)
    pred = model.predict(roi)
Convert Prediction to Text:
  label = decode_prediction(pred)
  print("Detected Sign:", label)
Convert Text to Speech:
  if voice_enabled:
    text_to_speech(label)
Display Output:
  cv2.imshow("Frame", frame)
  if cv2.waitKey(1) & 0xFF == ord('q'): break
Release Resources:
  cap.release()
  cv2.destroyAllWindows()
```

Model Architecture and Training

The proposed sign language recognition system is built upon the ResNet50 architecture as shown in table 4, a deep residual network pre-trained on the ImageNet dataset. To adapt the model for sign language classification, the top layers of the base model are replaced with a custom classification head. Specifically, a Global Average Pooling layer is applied to the output of the base model to reduce spatial dimensions, followed by a fully connected Dense layer consisting of 128 neurons with ReLU activation to learn high-level feature representations. A Dropout layer with a rate of 0.5 is incorporated to mitigate overfitting during training. Finally, a Softmax output layer is added, with the number of neurons corresponding to the total number of sign language classes, enabling multi-class probability estimation.

Table 3. ResNet50 Architecture Configuration

#	Layer	Configuration	Output Shape
1	Input Layer	224 × 224 × 3	(224, 224, 3)
2	ResNet50 Base	Pre-trained, Frozen, ImageNet weights	(7, 7, 2048)
3	Global Average Pooling 2D	Spatial averaging across feature maps	(2048,)
4	Dense Layer	128 units, ReLU activation	(128,)
5	Dropout Layer	Rate = 0.5	(128,)
6	Output Dense Layer	N units, Softmax activation	(N,)

Table 5. Hyperparameters Used for ResNet50 Model Training

#	Hyper-parameter	Value	#	Hyper-parameter	Value
1	Input Image Size	224 × 224 × 3	2	Batch Size	32
3	Number of Epochs	20	4	Validation Split	20% (0.2)
5	Optimizer	SGD	6	Learning Rate	0.001 (1e-3)
7	Momentum	0.9	8	Loss Function	Categorical Cross-Entropy
9	Evaluation Metric	Accuracy	10	Dropout Rate	0.5
11	Dense Layer Units	128	12	Activation (Hidden)	ReLU
13	Activation (Output)	Softmax	14	Pre-trained Weights	ImageNet
15	Base Model Layers	Frozen	16	Pooling Layer	Global Avg Pooling 2D

To leverage the rich feature representations learned from ImageNet, all layers of the ResNet50 base model are frozen during the initial training phase, ensuring that only the newly added classification layers are updated. Hyper parameters are listed in table5. Upon completion of training, the model's learning behavior is visualized through accuracy and loss curves plotted across epochs for both the training and validation sets. Upon successful completion of training, the optimized model is saved for subsequent deployment in the real-time sign language recognition pipeline.

Algorithm 2 Sign Language Gesture Recognition Training Using ResNet50

Input: Gesture image dataset
Output: Trained ResNet50 model
Data Preparation:
  Load gesture images and corresponding labels
  Resize images to (25×25) and normalize pixel values
  Encode labels into numerical format
  Split dataset into training and validation sets $(80\% : 20\%)$
  Store preprocessed dataset
Model Construction:
  Load pretrained ResNet50 without top layers
  Add Global Average Pooling layer
  Add Fully Connected layer with ReLU activation
  Add Dropout layer with $p = 0.5$
  Add Softmax output layer for classification
  Compile model using SGD optimizer and categorical cross-entropy loss
Training:
  Train model for $EPOCHS = 2$ with batch size $= 32$
  Save best model based on validation accuracy
Evaluation:
  Plot training and validation accuracy/loss curves
  Compute confusion matrix for performance analysis
Return: Trained model and evaluation metrics

Text-to-Speech Conversion

The system includes a Text-to-Speech (TTS) module. After a gesture is recognized, the predicted letter, or word, is displayed as text. The text is then sent to the TTS engine. The pyttsx3 speech synthesis library converts the text into speech. This allows hearing people to understand the recognized sign. By providing both text and speech output, the system improves communication between deaf and hearing individuals.

Convolution Neural Network Model Architecture and Training

The proposed sign language recognition system uses a custom designed Convolutional Neural Network (CNN) to classify hand gestures. The CNN architecture, shown in Figure 8, performs feature extraction and classification of gestures. The input to the CNN model is a pre-processed 256 × 256 × 1 grayscale gesture image. The first three convolution layers are used to extract features from gesture image. The first convolution layer applies 16 filters of size 2 × 2 with ReLU activation, followed by 2 × 2 max pooling. The second convolution layer uses 32 filters of size 5 × 5, followed by 5 × 5 max pooling. A third convolution layer with 64 filters of size 5 × 5 are further used. The resulting feature maps are then flattened into a one-dimensional feature vector and passed to a fully connected (dense) layer with 128 neurons using the ReLU activation function. A dropout layer with dropout rate of 0.2 is employed to reduce overfitting and improve the model's generalization capability. Finally, the output layer consists of N neurons, where N represents the 43 different gesture classes. Output layer uses the Softmax activation function to predict the probability of each of the 43 classes. The network is trained using the Cross-Entropy loss function with the Gradient Descent optimizer with the learning rate of 0.01. This architecture effectively extracts spatial features from hand gesture images and performs accurate multi-class classification for sign language recognition.

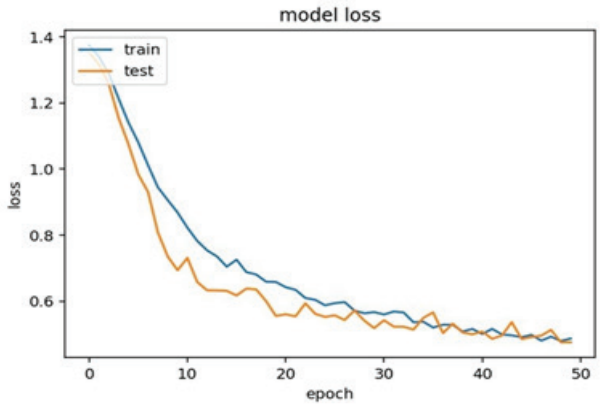


Fig 3. Training Results

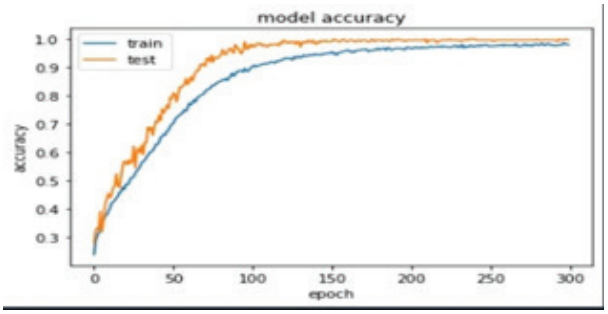


Fig 4. Training Results

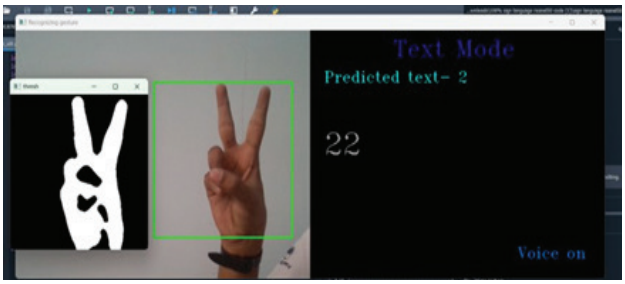


Fig 5. Output

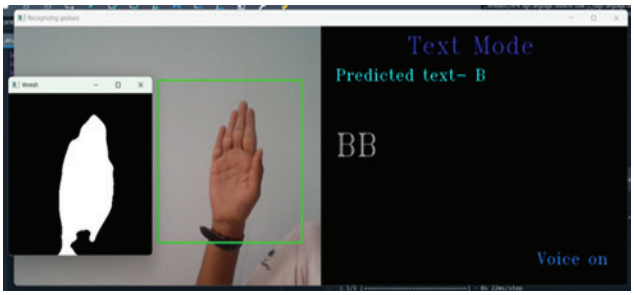


Fig 6. Output

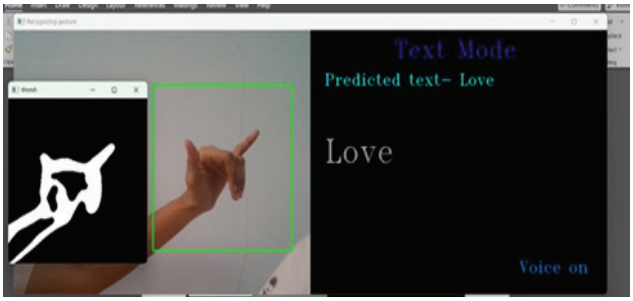


Fig 7. Output

An 80:20 train-validation split is used in this implementation to ensure balanced distribution. Data augmentation techniques such as rotation, flipping, contrast adjustment improve performance and generalization ability of the model for the real-world variations. The network is trained using the categorical cross-entropy loss function and optimized using the Adam optimizer with a learning rate of 0.001, ensuring stable and fast convergence. Mini-batch gradient descent is employed with a batch size of 32, and the model is trained for 10 epochs. To further improve robustness, early stopping monitors validation accuracy and halts training when no significant performance gain is observed. The model weights are initialized using He initialization, which facilitates stable gradient propagation across deeper layers.

RESULTS

Experimental Setup

The two deep learning models : ResNet50 and custom CNN model are implemented and trained on a high-performance AI workstation featuring a 14th Gen Intel® Core™ i9-14900K processor with 64 GB DDR5 RAM, and an NVIDIA® GeForce RTX™ 4090 GPU with 24 GB GDDR6X memory. The powerful hardware configuration enabled reduced the training time.



Fig 8. Convolutional Neural Network Model Architecture

Algorithm 3 Real-Time Gesture Recognition Using Convolutional Neural Network

```
Load the Model:
    Load pretrained CNN model for gesture classification
Initialize Video Capture:
    Start device camera for real-time video input
Define Detection Box:
    Draw rectangular region for hand placement
Extract Region of Interest (ROI):
    Capture image inside detection box
Preprocess ROI:
  ROI = resize(ROI, (64, 64))
  ROI = normalize(ROI)
  ROI = reshape(ROI)
Make Predictions:
  prediction, confidence = model.predict(ROI)
Confidence Filtering:
  if confidence > 0.9:
    accept_prediction()
  else:
    discard_prediction()
Display and Convert Gesture:
    Display recognized letter on screen
    text_to_speech(prediction)
Exit and Release Resources:
    Stop video capture
    Release system resources
```

Output

Figure 3 and Figure 4 illustrates the ResNet50 model's loss and accuracy over 50 epochs for both training and testing datasets. Figure 5, 6 and 7 show the system output for ResNet50 model, where the hand gesture is detected and classified in real time. The detected hand gesture is

shown in closed green bounding box. The recognized letter is displayed on the screen. GUI guides the user to position their hand correctly for accurate detection. The predicted text is displayed on the screen, and the voice output is also generated, making the system interactive and accessible.

Figure 12 demonstrates model training log of the proposed CNN model. Figure 9 and Figure 10 illustrates the custom CNN model's loss and accuracy results. Figure 11 show the system output for CNN model, where the hand gesture is detected and classified in real time.

Comparison of Model Performance

Table 6. Performance Results

Model	Accuracy (%)
ResNet50	94.5
CNN	88.2

Table 6 shows a comparative study of different algorithms for sign language recognition. Among the models, ResNet50 gives a high accuracy (94.5%) with lower complexity compared to the deep learning model with custom CNN. ResNet50 is preferred for ef4.4. Evaluation Metrics

In sign language recognition, evaluating model performance is essential to ensure accurate translation of gestures into text. The most commonly used metrics are Precision, Recall, Accuracy, and F1-Score.

Figure 13 illustrates the confusion matrix of the proposed CNN model for recognizing 43 sign language classes. The matrix exhibits a strong diagonal pattern, indicating that the majority of test samples are correctly classified into their respective classes. The dark blue diagonal cells represent high true positive predictions.

CONCLUSION

This paper presented a sign language recognition system that converts static hand gestures for 43 different classes including alphabets, numbers and few commonly used symbolic gestures into text and speech using deep learning. Two models, a custom CNN and the pre-trained ResNet50, were implemented and evaluated. Experimental results showed that the custom CNN achieved an accuracy of 88.2%, while ResNet50 achieved 94% accuracy. The higher performance of ResNet50 demonstrates the advantage of transfer learning for extracting meaningful features from hand gesture images. By providing real-time text and speech output, the system has the potential to improve communication between deaf or mute individuals and people who are not familiar with sign language. Future improvements include integrating deep learning for dynamic recognition, optimizing real-time processing, and incorporating facial expressions and body posture for enhanced accuracy and usability, efficient and accurate performance.

Table 7. Performance Evaluation Metrics

Metric	Definition	Formula
Accuracy	Percentage of correctly classified samples among all samples.	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	Percentage of predicted positive samples that are actually positive.	$\frac{TP}{TP+FP}$
Recall	Percentage of actual positive samples correctly identified by the model.	$\frac{TP}{TP+FN}$
Confusion Matrix	A table that summarizes the classification results using true and predicted classes.	TP - True Positive , TN- True Negative, FP- False Positive, FN- False Negative.

Table 8. Confusion Matrix

Actual	Predicted	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

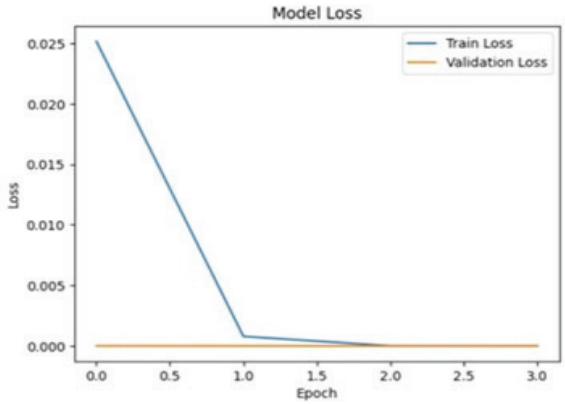


Fig 10. Training Results

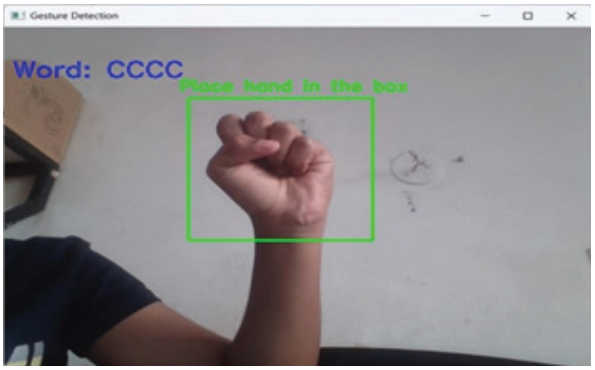


Fig 11. Experimental Result

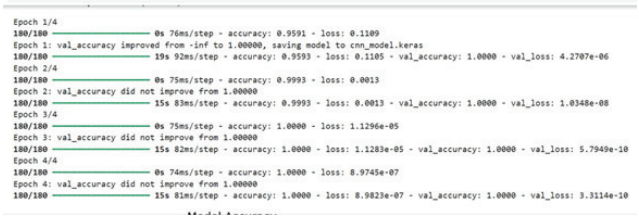


Fig 12. Training Epoch

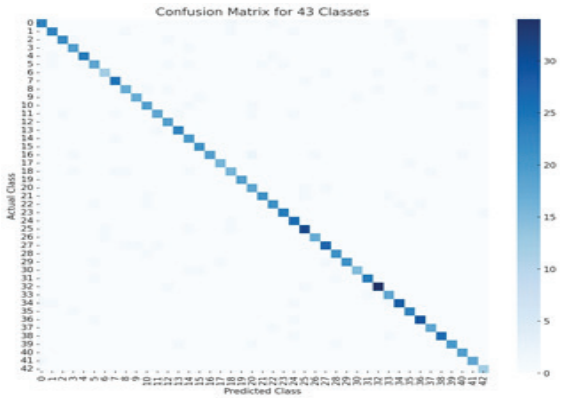


Fig 13. Confusion Matrix

REFERENCES

1. N. M. et al., "A review of the hand gesture recognition system: Current progress and future directions," IEEE Access, vol. 9, pp. 157422–157433, 2021.
2. G. I. Haidar and H. I. Reefat, "Glove-based american sign language interpretation using cnn and data glass," in Proc. IEEE Int. Conf. on Artificial Intelligence, 2020.
3. Y. Y. et al., "Research on dynamic sign language algorithm based on sign language trajectory and key frame extraction," IEEE Trans. Image Process., vol. 28, no. 3, pp. 1–10, 2019.
4. D. P. et al., "A deep learning-based approach for hand sign recognition using cnn architecture," IEEE Access, vol. 9, pp. 115322–115335, 2021.
5. H. S. L. Vaishnavi D. A., A. Kumar C. and D. M. L., "Mediapipe to recognise the hand gestures," WSEAS Trans. Signal Process., vol. 18, pp. 134–140, 2022.
6. B. N. et al., "A survey paper on hand gesture recognition," Int. J. Comput. Vision Signal Process., vol. 8, no. 2, pp. 45–55, 2020.
7. A. A. B. et al., "Cnn-based feature extraction and classification for sign language," in Proc. IEEE Int. Conf. on Computer Vision, 2021.
8. S. T. et al., "Sign language recognition and converting into text," J. Comput. Sci. Appl., vol. 12, no. 4, pp. 67–75, 2019.
9. Y. Y. et al., "Video hand gesture recognition using depth camera and lightweight cnn," in Proc. IEEE Int. Conf. on Artificial Intelligence and Machine Learning, 2022.
10. S. T. et al., "Sign language to text conversion in real-time using transfer learning," in Proc. IEEE Int. Conf. on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 345–350.
11. P. Duraisamy et al., "Transforming sign language into text and speech through deep learning technologies," SRS Journal, December 2023, accessed 2024.
12. M. Munde et al., "A real-time sign language to text conversion system for enhanced communication accessibility," Journals.tultech.eu, August 2024, accessed 2024
13. T. Fathima et al., "Real-time sign language recognition and translation using deep learning techniques," Irjaeh, 2024, accessed 2024.
14. E. Stergiopoulou and N. Papamarkos, "A new technique

for hand gesture recognition (sgong method)," Pattern Recognition Letters, vol. 42, pp. 1–9, 2019

15. M. K. S. et al., "Hand gesture recognition with depth images: A review," in Proc. IEEE Int. Conf. on Image Processing and Machine Vision, 2020
16. E. A. et al., "Real-time translation of indian sign language using lstm," in Proc. Int. Conf. on Neural Networks and Signal Processing, 2020
17. K. Murakami and H. Taguchi, "Gesture recognition using recurrent neural networks," Fujitsu Laboratories Ltd., Kawasaki, Japan, Tech. Rep., 2020.
18. B. X. et al., "Application of tensor train decomposition in s2vt model for sign language recognition," IEEE Trans. Signal Process., vol. 68, pp. 1124–1135, 2021.
19. H. Garg et al., "Deep learning-based sign language translator," in Springer Lecture Notes, July 2024, accessed 2024.
20. A. Almjally and W. S. Almukadi, "Deep computer vision with artificial intelligence based sign language recognition to assist hearing and speech-impaired individuals," Scientific Reports, vol. 15, no. 1, pp. 1–15, 2025.
21. M. Balat, R. Awaad, A. B. Zaky, and S. A. Aly, "Revolutionizing communication with deep learning and xai for enhanced arabic sign language recognition," arXiv preprint arXiv:2501.08169, 2025.
22. B. Alabduallah and A. A. Alneil, "Revolutionizing sign language recognition for hearing-impaired persons using ensemble of deep learning techniques with fine tuning model," Journal of King Saud University - Computer and Information Sciences, vol. 38, no. 2, 2026. [Online]. Available: https://link.springer.com/article/10.1007/ s44443-025-00395-7
23. A. I. Rastogi, A. I. Mahapatra et al., "Advanced gesture recognition in Indian Sign Language using a synergistic combination of YOLOv10 with Swin Transformer model," Scientific Reports, vol. 15, p. 32894, 2025. [Online]. Available: https://www.nature.com/articles/s41598-025-18496-8
24. J. Hao and H. Pan, "A novel deep transformer based cvt model for sign language recognition in visual communication," Scientific Reports, vol. 16, pp. 1–15, 2025.
25. H. Harrouch, L. Trabelsi, M. Jebali, and O. Gammoudi, "A deep learning-based method combines manual and non-manual features for sign language recognition," Scientific Reports, vol. 16, no. 1, p. 2981, 2025.
26. V. K. Gajjala, N. T. J., L. R. Battarusetty, and V. Avula, "A vision-based sign language recognition & voice conversion system for inclusive communication," Scientific Reports, vol. 16, no. 1, p. 35541, 2026. [Online]. Available: https://www.nature.com/articles/s41598-026-35541-2
27. J. K. Nath, S. A. M. Alam, Y. Ullah, and Z. H. Abdullah, "Visual deep learning approaches for alphabetic sign language interpretation," Scientific Reports, vol. 16, no. 1, p. 57945, June 2026.
28. R. Gupta and S. R. Nalla, "Efficientsign: An attention-enhanced lightweight architecture for indian sign language recognition," 2026. [Online]. Available: https://arxiv.org/abs/2604.08694

Table 2. Comparison of Recent Deep Learning-Based Sign Language Recognition Methods (2025–2026)

Research Paper	Model	Dataset	Acc.	Prec.	Rec.	F1	Limitations
Almjally et al. (2025) [20]	ResNet-152 + BiLSTM + Harris Hawk Optimization	Kaggle Sign Language for Numbers (15,000 images, 10 classes)	98.95	94.72	94.74	94.72	Evaluated only on static number gestures; continuous recognition and text-to-speech translation were not considered.
Mazen Balat et al. (2025) [21]	ResNet50, MobileNetV3, EfficientNet-B2 + XAI	ArSL2018, AASL	99.48	NR	NR	NR	Limited to Arabic Sign Language datasets.

Alabduallah et al.(2026) [22]	InceptionV3 + BiLSTM + GRU + VAE + Tasmanian Devil Optimization	American Sign Language Dataset	98.18	76.77	76.52	76.45	High computational complexity; evaluated only for isolated ASL gestures.
Rastogi et al. (2025) [23]	YOLOv10 + Swin Transformer + Mish Activation	Custom Indian Sign Language Dataset (15,000 images + 35 videos)	97.62				
(mAP)	97.50	98.10	96.58	Small dynamic video dataset; focuses on gesture recognition rather than complete translation.			
Hao and Pan (2025) [24]	Convolutional Vision Transformer (CvT)	Sign Language Digits Dataset; ASL Alphabet Dataset	99.00	NR	NR	NR	Evaluated only on isolated digits and alphabet datasets.
Haifa Harrouch et al. (2026) [25]	Hybrid Deep Learning Model	Public Sign Language Dataset	NR	NR	NR	NR	Higher computational complexity due to multimodal feature extraction.
Gajjala et al. (2026) [26]	CNN + Transformer	Custom Gesture Dataset	99.00	NR	97.00	97.00	Limited vocabulary and language-specific evaluation.
Joyonta Kumar Nath et al. (2026) [27]	ConvNeXtXLarge, EfficientNet, VGG19, ResNet50	ASL Alphabet Dataset	99.81	NR	NR	NR	Limited to alphabet-level recognition.
Gupta et al. (2026) [28]	EfficientNet-B0 + Attention	Indian Sign Language Alphabet Dataset (12,637 images)	99.94	NR	NR	NR	Evaluated only on alphabet-level Indian Sign Language.