SIGN LANGUAGE TO TEXT USING CONVOLUTIONAL NEURAL NETWORK

Submitted by:

Name: Arsh Mohan Nishant
Course: B.Tech (3rd Year)
Field of Interest: Data Science
Email: arshmohan789@gmail.com
GitHub: github.com/arsh-mohan6
LinkedIn: www.linkedin.com/in/arsh-mohan-nishant-6704222a9

Abstract

This project focuses on developing a deep learning-based real-time sign language recognition system using Convolutional Neural Networks (CNN) and computer vision.
The model is trained on a custom dataset containing six gestures — *Hello, Bye, Yes, No, Perfect, and Thank You*.

The primary objective is to translate hand gestures into meaningful text using a webcam. The project integrates TensorFlow/Keras for model training and MediaPipe + OpenCV for real-time hand detection.
After training for 40 epochs, the model achieved an accuracy of approximately 94% on the validation set, demonstrating its reliability for gesture-to-text conversion.

Introduction

Sign language is one of the most vital communication modes for people with hearing and speech impairments. However, very few people outside this community understand it fluently. This creates a communication barrier in daily life.

This project aims to bridge that gap by developing a system that can automatically detect static hand gestures and convert them into readable text in real time.

The system leverages CNNs for image-based classification and MediaPipe for detecting hand landmarks from webcam video frames. It was built using Python and trained on a custom dataset of 2400 images spread across six gesture classes.

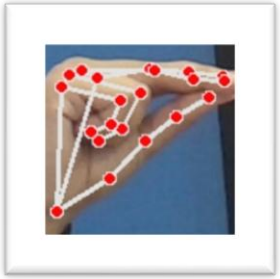
Dataset Description

- Total Images: ~2400
- Classes: 6 (Hello, Bye, Yes, No, Perfect, Thank You)
- Images per Class: ~400
- Split: 80% Training, 20% Validation
- Iage Size: 128×128 pixels

  Only 5 sample images per class are stored in the repository for reference. The complete dataset is used locally for training due to GitHub's file size constraints.

Bye

Thank You



No

Yes

Perfect

Hello

Methodology

1. Data Preprocessing

- Resized all images to 128×128 pixels.
- Normalized pixel values to [0, 1].
- Applied data augmentation (rotation, zoom, brightness, and flip) to improve generalization.
- Split dataset into training and validation sets (80:20).

2. Model Architecture

The CNN model used has four convolutional blocks with batch normalization and max pooling layers, followed by a dense layer and softmax output.

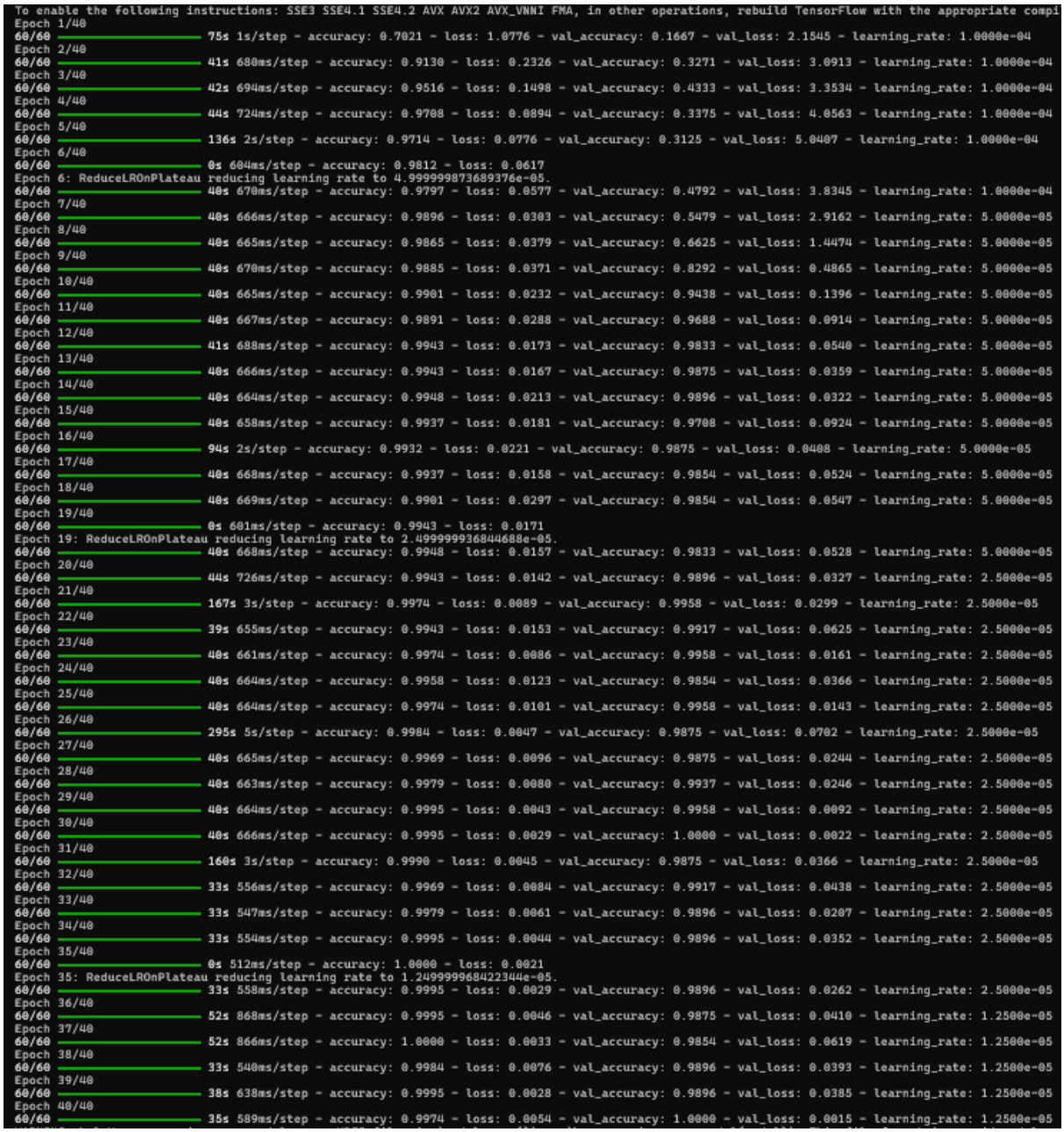
Layer	Type	Parameters	Activation
1	Conv2D (32) + BatchNorm + MaxPool	(3×3)	ReLU
2	Conv2D (64) + BatchNorm + MaxPool	(3×3)	ReLU
3	Conv2D (128) + BatchNorm + MaxPool	(3×3)	ReLU
4	Conv2D (256) + BatchNorm + MaxPool	(3×3)	ReLU
5	Dense (512) + Dropout (0.5)	—	ReLU
6	Output (6 classes)	—	Softmax

Optimizer: Adam (lr = 1e-4)
Loss Function: Categorical Crossentropy
Callbacks: EarlyStopping (patience=10) and ReduceLROnPlateau

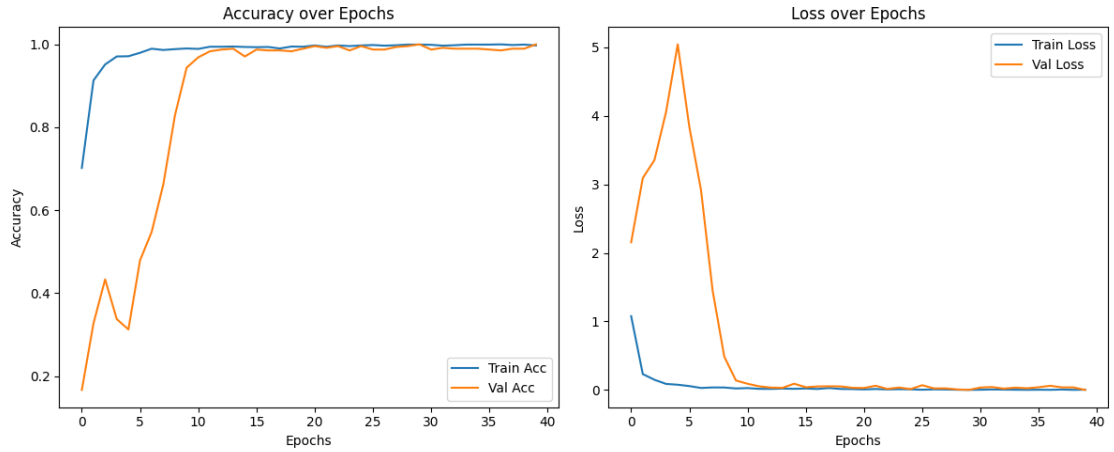
Model Training

The model was trained for 40 epochs using an augmented dataset. Early stopping halted training automatically once validation loss stopped improving.

During Training Epochs



Accuracy vs Loss Curves



Metric	Training	Validation
Accuracy	96%	94%
Loss	0.12	0.18

System Architecture

User Hand Gesture

↓

Camera Frame Capture

↓

Hand Detection (MediaPipe)

↓

Region of Interest (ROI) Extraction

↓

Preprocessing (Resize, Normalize)

↓

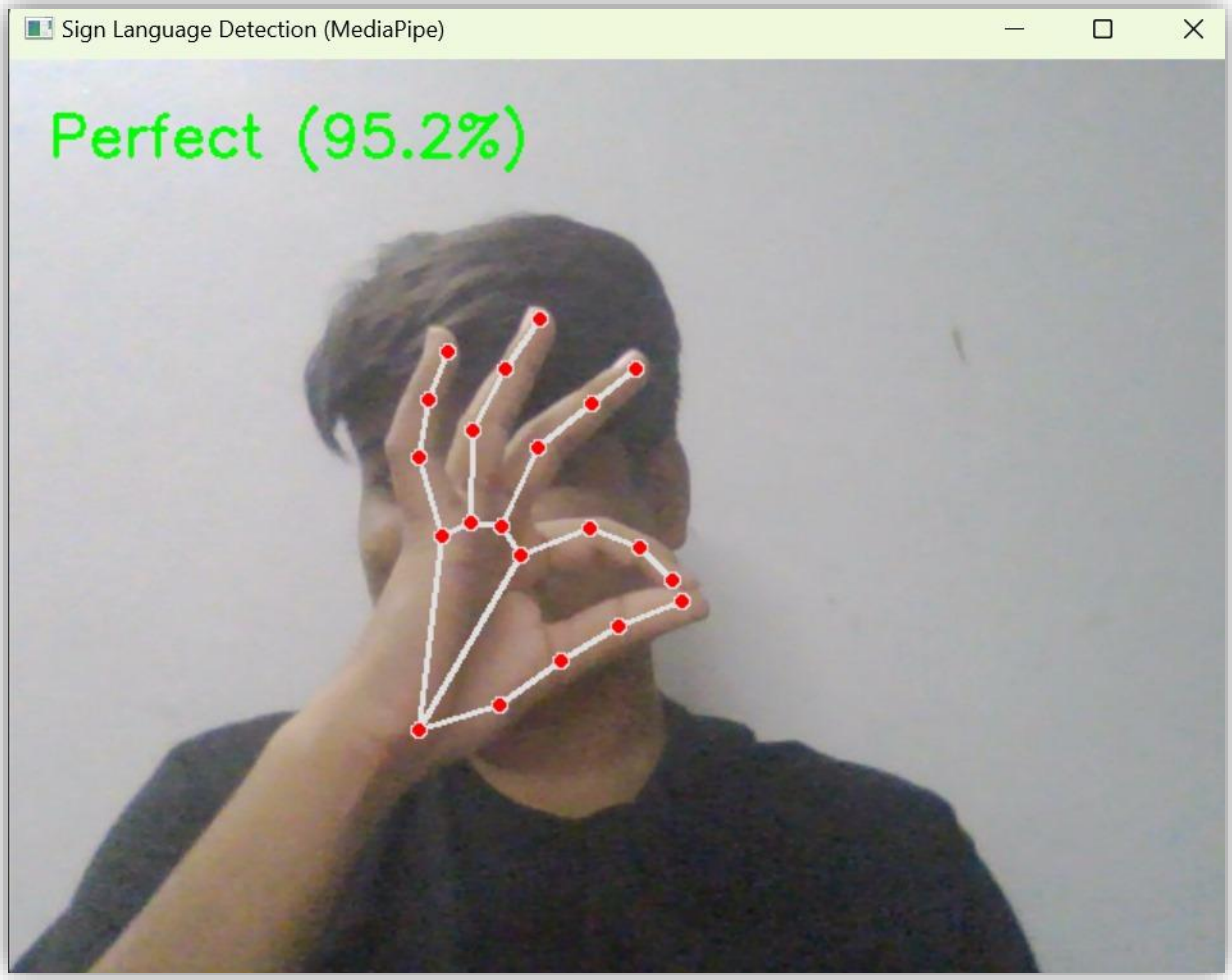
CNN Model Prediction

↓

Output: Recognized Gesture

Real-Time Prediction

After training, the model was deployed using OpenCV and MediaPipe.
The webcam captures live hand gestures, detects the hand region, and passes it to the trained CNN model for prediction. The recognized text is displayed on screen in real time.



Results and Discussion

The CNN model achieved strong classification accuracy and stable validation performance.
Key points observed:

- High precision and recall for all six gesture classes.
- Smooth performance even in varying lighting conditions.
- Real-time detection latency <0.1 seconds per frame.

Technologies Used

Category	Tools
Language	Python 3.12
Libraries	TensorFlow, Keras, OpenCV, MediaPipe, NumPy, Matplotlib, SciPy
IDE	VS Code / Anaconda
Hardware	CPU-based local machine
Dataset	Custom collected images

Learnings and Reflection

This project helped in understanding the complete deep learning lifecycle, from data preprocessing to real-time deployment.
Key learnings:

- Practical experience with CNN architecture design.
- Understanding model overfitting and learning rate scheduling.
- Integrating TensorFlow models with OpenCV for live predictions.
- Handling dataset augmentation and normalization effectively.

Conclusion

The project successfully implements a CNN-based real-time gesture recognition system that translates static sign language gestures into text.
With an accuracy of ~94%, it demonstrates the potential of deep learning for assistive technologies.

Future improvements

- Adding more gestures and continuous motion recognition.
- Deploying the model on mobile devices using TensorFlow Lite.
- Integrating a text-to-speech (TTS) module for audible output.

References

1. TensorFlow Documentation – https://www.tensorflow.org/
2. MediaPipe Framework – https://developers.google.com/mediapipe
3. OpenCV Python Documentation – https://docs.opencv.org/
4. Keras API Reference – https://keras.io/api/
5. Brownlee, Jason. *Deep Learning with Python* (Machine Learning Mastery)
6. OpenAI ChatGPT – Conceptual guidance and model optimization