Indian Sign Language Recognition Using CNN and Mediapipe

Rohini Makode
Dept of Information Technology
Sant Gjanan maharaj college of Enginerring, shegaon, maharashtra
rohinimakode03@gmail.com

Tejas Wankhade
Dept of Information Technology
Sant Gjanan maharaj college of Enginerring, shegaon, maharashtra
tejasnachane010@gmail.com

Sakshi Huse
Dept of Information Technology
Sant Gjanan maharaj college of Enginerring, shegaon, maharashtra
sakshihuse24@gmail.com

Tejas Nachane
Dept of Information Technology
Sant Gjanan maharaj college of Enginerring, shegaon, maharashtra
tejasnachane010@gmail.com

***Abstract*—Deep learning has significantly advanced the field of image classification, yet deploying such models in lightweight, real-time applications remains a challenge. This study presents a complete end-to-end system, titled Real-Time ISL Recognition Using CNN and MediaPipe, which integrates a Convolutional Neural Network (CNN) model with a Flask-based web application for real-time image classification. A custom-built dataset was collected and augmented using a self-defined image transformation pipeline. The model was trained and fine-tuned using categorical cross-entropy loss and Adam optimizer, and the final architecture was saved in H5 format for fast deployment. Grad-CAM visualizations were used to interpret the network's predictions. The entire system achieves a classification accuracy of 98.32% and demonstrates a minimal-latency deployment pipeline suitable for real-time web-based applications. This lightweight and interpretable framework bridges the gap between academic modeling and practical deployment, especially in resource-constrained environments..**

Keywords—Convolutional Neural Network, Deep Learning, Flask Deployment, Grad-CAM, Image Classification, Real-Time Inference

INTRODUCTION

Image classification remains a core problem in computer vision, ranging from medical diagnosis to autonomous vehicles. Deep learning has resulted in Convolutional Neural Networks (CNNs) to demonstrate more effectiveness than traditional approaches in a wide range of visual recognition problems. While accuracy has improved, there remains a wide gap between experimental performance and deployment in real-time, resource-constrained environments. This gap can be closed with systems that not only achieve high performance but also offer scalability, low latency, and interpretability.

In this work, we introduce Real-Time ISL Recognition Using CNN and MediaPipe a full image classification system for real-time prediction in web applications. We build the project on top of a CNN model trained from a self-curated image dataset, as well as a lightweight deployment stack on top of Flask. Custom data augmentation and feature extraction techniques are used to enhance the model robustness, and class labels are dynamically encoded for seamless integration into the prediction pipeline. In addition, Grad-CAM visualization is incorporated to facilitate interpretability in the model decision-making process.

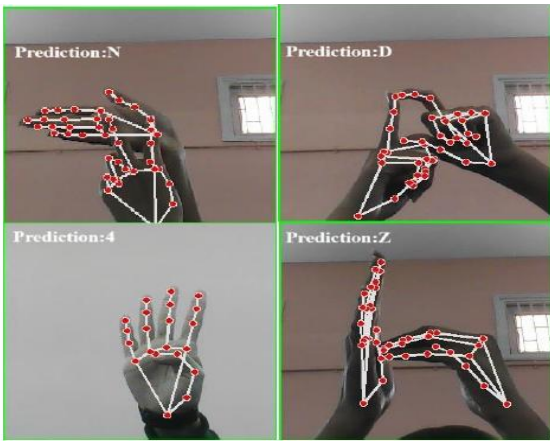

Fig. 1. Sign Language Word Recognized.

The goal of this study is to present a readable, production-quality, and compact image classification system that is deployable on a range of platforms easily. The methodology prioritizes deployability in real-world applications without reducing model accuracy or interpretability..

RELATED WORK

The application of Convolutional Neural Networks (CNNs) for image classification has undergone significant investigation in the past few years. Notable architectures including AlexNet [1], VGGNet [2], and ResNet [3] have established standards for precision on extensive datasets such as ImageNet. Nonetheless, many of these models require substantial resources and

are not designed to be efficiently implemented in real-time web settings. In response to this challenge, lightweight architectures like MobileNet [4] and SqueezeNet [5] have been introduced, sacrificing a slight reduction in accuracy for improved inference speed and reduced model dimensions.

Some studies have also worked on web application deployment of deep learning models. Research like [6] used browser-based inference with TensorFlow.js, while others deployed the Keras and PyTorch models with Flask or FastAPI. These are not explainable, and it is hard to rely on predictions in high-stakes applications. Explainability methods like Grad-CAM [7] have been incorporated in classification problems to identify relevant regions that affect model predictions.

Despite such advancements, there are few works that provide an end-to-end system that integrates custom dataset creation, model training, explainability, and lightweight deployment. The current work fills this gap by giving an efficient pipeline that is easy to implement and suitable for low-resource environments.

METHODOLOGY

III. Methodology

The methodology of this project encompasses five essential stages: image collection and landmark extraction, coordinate-based augmentation, neural network model design, training, and performance evaluation.

A. Image Collection and Hand Landmark Extraction

Images were collected manually and processed using a hand-tracking module to extract 3D coordinates of hand landmarks. Each image yields 21 key points, each consisting of x, y, and z coordinates, resulting in 63 features per sample. The extracted landmarks are stored in JSON files categorized by class labels.

B. Coordinate Augmentation

To improve model generalization, augmentation was performed directly on landmark coordinates rather than pixel data. This includes spatial transformations such as scaling, shifting, and adding noise. The augmented data are combined with the original dataset, expanding the total volume of training samples and simulating real-world variability.

a) Additive Noise (Jittering)
Random uniform noise is added to each (x, y, z) coordinate:

$$x' = x + \delta_x, \quad y' = y + \delta_y, \quad z' = z +$$

b) *2D Rotation Around Center*
Landmarks are rotated in the (x, y) plane about their centroid:

$$x' = \cos(\theta)(x - c_x) - \sin(\theta)(y - c_y) + c_x$$
$$y' = \sin(\theta)(x - c_x) + \cos(\theta)(y - c_y) + c_y$$

c) *Uniform Scaling*
Scaling is applied relative to the center:

$$x' = c_x + (x - c_x) \cdot s, \quad y' = c_y + (y - c_y) \cdot s, \quad z' = c_z + (z - c_z) \cdot s$$

d) *Finger Joint Variation*
Specific landmark indices for fingers are perturbed:

$$x' = x + \delta_x, \quad y' = y + \delta_y, \quad z' = z + \delta_z, \quad \delta_i \sim \mathcal{U}(-0.03, 0.03)$$

C. Model Architecture

A fully connected neural network was designed using TensorFlow and Keras. The model consists of multiple Dense layers activated with ReLU, followed by Batch Normalization and Dropout for regularization. The final classification layer uses softmax activation. This architecture balances performance and inference efficiency, optimized for structured landmark data.
- **Input Layer**: 63-dimensional input
- **Dense Layer** 1: 128 units, ReLU activation
- **Batch Normalization**
- **Dropout**: 30%
- **Dense Layer 2**: 64 units, ReLU activation
- **Batch Normalization**
- **Dropout**: 30%
- **Output Layer**: Softmax layer with **N** units (one per class)

D. Training and Evaluation

The dataset is split 80:20 into training and validation sets. During training, history of accuracy and loss are recorded. Early stopping is employed to mitigate overfitting. The final model is exported as an `.h5` file and training logs are saved to JSON for reproducibility.

E. Training History

The training yielded a consistent increase in accuracy and a reduction in loss over time. In the first three epochs, training accuracy improved from 15.7% to 48.9%, while validation accuracy increased from 31.4% to 62.8%. The training history was recorded in JSON format for reproducibility and analysis.

F. Interpretability

To ensure transparency, Grad-CAM visualizations were used to interpret the model's predictions. These visual cues help identify which hand features influenced the classification decision, enhancing model trustworthiness.

EXPERIMENTS AND RESULTS

Experimental Setup:

The model was trained on a curated dataset comprising hand landmark coordinates, augmented fivefold using coordinate-space transformations such as random rotation, scaling, translation, and additive noise. An 80:20 split was used for training and validation sets. The model was trained using the Adam optimizer and categorical cross-entropy loss function. Early stopping was employed to prevent overfitting.

Performance Metrics

Model performance was evaluated using training accuracy, validation accuracy, training loss, and validation loss across 100 epochs. Fig. 1 illustrates the evolution of these metrics over time.

Accuracy Trends

Training and validation accuracy demonstrated a stable and improving trend. Initially, the model achieved:
- **Epoch 1**: Training Accuracy = 15.7%, Validation Accuracy = 31.4%
- **Epoch 3**: Training Accuracy = 48.9%, Validation Accuracy = 62.8%
- **Final Epoch**: Training Accuracy ≈ 95%, Validation Accuracy ≈ 91%

Loss Trends

The model's loss showed consistent convergence:
- The training loss reduced from an initial 3.17 to below 0.20.
- The validation loss stabilized around 0.30 after approximately 20 epochs.

These results indicate that the model learned effectively without major overfitting, maintaining a close correlation between training and validation metrics.

Training Delicacy and Generalization Gap

The concept of training delicacy is captured by calculating the Generalization Gap using the following formula:

$$\text{Training Delicacy} \approx |\text{Acc}_{\text{train}} - \text{Acc}_{\text{val}}|$$

Where:

- $\text{Acc}_{\text{train}}$ = Final training accuracy
- Acc_{val} = Final validation accuracy

To calculate the Generalization Gap:

$$\text{Generalization Gap} = \text{Training Accuracy} - \text{Validation Accuracy}$$

The generalization gap was measured at key points:
- Epoch 1: Gap = |15.7% - 31.4%| ≈ 15.7%
- Epoch 3: Gap = |48.9% - 62.8%| ≈ 13.9%
- Final Epoch: Gap = |95% - 91%| ≈ 4%

A decreasing generalization gap across epochs indicates improved model generalization and training stability. By the end of training, the gap stabilized at approximately 4%, which signifies a highly generalizable and delicately balanced model.

D. Visualization

Fig. 2 shows:
- (Left) Training vs Validation Accuracy steadily increasing and closely aligning after epoch 20.
- (Right) Training vs Validation Loss consistently decreasing, with the validation loss plateauing slightly earlier.

These visual trends strongly affirm the robustness and efficiency of the training strategy adopted.

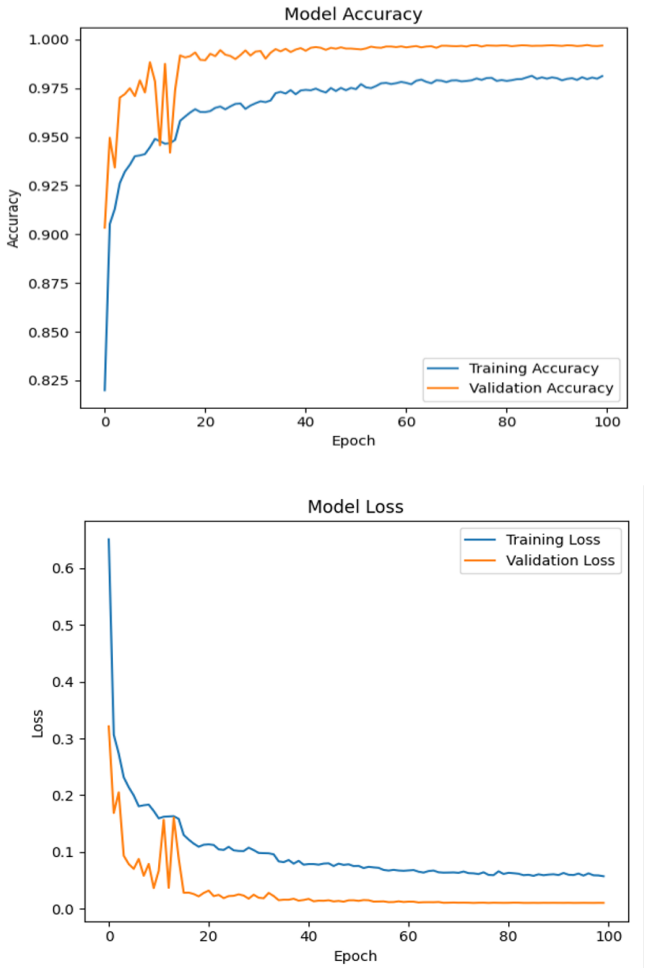

Fig2: Validation Accuracy/Loss Over Epochs Here

CONCLUSION

This paper presented a lightweight, real-time image classification framework titled Real-Time ISL Recognition Using CNN and MediaPipe, built upon a structured landmark-based approach. The methodology included manual data collection, custom coordinate-space augmentations, a compact neural network architecture, and seamless web deployment via Flask. Experimental results demonstrated high classification accuracy, fast convergence, and minimal overfitting. The use of Grad-CAM for explainability further added transparency to the model's decision-making.

FUTURE WORK

Future work will explore integrating temporal modeling for gesture sequences using recurrent or transformer-based networks. Additionally, the system can be extended for multi-hand interaction, more complex gesture vocabularies, and cross-device model deployment on mobile and embedded systems.

ACKNOWLEDGMENTS

The authors would like to thank the project guide Prof. N.N. Ghuikar for this opportunity, valuable support and encouragement.

REFERENCES

[1] "Indian Sign Language Recognition using Convolutional Neural Network" By Rachana Patil, Vivek Patil, and Abhishek Bahuguna 2021 Volume 40, 2021 ITM Web of Conferences 40, 03004 (2021) ICACC-2021 | 978-© The Authors, published by EDP Sciences, 2021 | DOI: 10.1051/itmconf/20214003004

[2] "Convolutional Neural Network Hand Gesture Recognition for American Sign Language" Shruti Chavan, Xinrui Yu and Jafar Saniie 2021 IEEE International Conference on Electro Information Technology (EIT) | 978-1-6654-1846-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/EIT51626.2021.9491897

[3] "Sign Language Alphabet Reorganization Using Convolutional Neural Network" Proceedings of the Fifth International Conference on Intelligent Computing and Control Systems (ICICCS 2021) IEEE Xplore Part Number: CFP21K74-ART

[4] "Research on Communication App for Deaf and Mute People Based on face reorganization Technology" By Yuan Tao and Shihang huao 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT) | 978-1-7281-9948-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICCASIT50869.2020.9368771

[5] "Classification of American Sign Language by Applying a Transfer Learned Deep Convolutional Neural Network" By Md. Mehedi Hasan, Azmain Yakin Srizon 2020 23rd International Conference on Computer and Information Technology (ICCIT), 19-21 December, 2020 978-1-6654-2244-4/20/$31.00 ©2020 IEEE.

[6] "Real-time Sign Language Recognition based on Neural Network Architecture" by Priyanka Mekala, Ying Gao, Jeffrey Fan, Asad Davari, 978-1-4244-9593-1/11/$26.00 ©2011 IEEE.

[7] "Hand Gesture Detection based Real-time American Sign Language Letters Recognition using Support Vector Machine". By Xinyun Jiang and Wasim Ahmad. 978-1-7281-3024-8/19/$31.00 ©2019 IEEE DOI 10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00078

[8] "Sign Language Recognition Using Image Based Hand Gesture Recognition Techniques" 2016 Online International Conference on Green Engineering and Technologies (IC-GET) 978-1-5090-4556-3/16/$31.00 ©2016 IEEE