# *Indian Sign Language Recognition Using Custom CNN with PyTorch Lightning*

---


## Abstract

The adding of sign language recognition definitely represents the bridge that brings together communication between the hearing and speech-impaired community and the rest of the society. This study proposes deep-learning-based work in classifying Indian Sign Language (ISL) gestures by using a customized Convolutional Neural Network (CNN) built with PyTorch Lightning. The model achieves a test accuracy of 86.52% on 23 classes. The pipeline for data, architecture, training methods, and evaluation are all described in this paper, demonstrating the workability of lightweight CNNs for real-time sign recognition tasks.


---

## 1. Introduction

Sign language recognition is one of the most important things in inclusive communication systems. In terms of traditional methods, they largely relied on sensor gloves or removal from the background. Their modern alternatives, however, rely entirely on vision-based deep learning, which offers greater improvement in scalability and deployment. Compared with the rest of the world's sign languages, Indian Sign Language appears to have gathered less attention from the research community. The paper presents a simple but efficient custom CNN-based ISL recognition system implemented using PyTorch Lightning. This paper aims to achieve a fair trade-off between the performance and model complexity.

---

## 2. Related Work

Numerous investigations have attempted to find different ways of applying deep learning to the recognition of gestures and sign languages. These studies include: Simonyan and Zisserman (2014) who introduced very deep convolutional networks for visual recognition【10】. Molchanov et al. (2016), who applied 3D CNNs for the recognition of gestures【1】. Kumar et al. (2020), who devised ISL recognition systems via transfer learning on MobileNetV2【2】. Singh et al. (2022), who realized high accuracies in static ISL classification through CNN application【3】. We, in turn, propose a custom-built CNN using PyTorch Lightning that emphasizes modularity and reproducibility.

---

## 3. Methodology

### 3.1 Dataset

In the creation of an open-access ISL alphabet dataset with 23 classes (A–Z excluded H, Y, and J), the dataset is arranged in class-specific subfolders and is preprocessed using torchvision.transforms (rotation, resizing, random horizontal-flipping, and normalization).

## 3.2 Data Module

We implemented a PyTorch Lightning DataModule to handle the training and testing splits (80:20), the transformations, and batch loading. Every image is resized and normalized to suit ResNet conventions【4】.

## 3.3 Model Architecture

Three convolutional layers are followed by two fully connected layers in the dense layers of our the touch-and-try system:

• Conv1: 3x3, 6 filters

• Conv2: 3x3, 16 filters

• FC1: 165454 → 120

• FC2: 120 → 84

• FC3: 84 → 20

• Output Layer: 20 → 23 (softmax)

Each of the convolution layers has a ReLU activation followed by max-pooling.

## 3.4 Training Strategy

The model has been optimized for 100 epochs using the Adam optimizer at a learning rate of 0.001. The main loss function used was cross-entropy, while accuracy metrics were the monitoring indicators for model evaluation during training and testing phases. The management of experimentations was greatly enhanced through checkpointing and the trainer API from PyTorch Lightning【5】.

---

# 4. Results and Evaluation

## 4.1 Accuracy

The final model achieved:

- **Test Accuracy:** 86.52%
- **Test Loss:** 1.0159

## 4.2 Classification Report

Precision, recall, and F1-scores were computed per class. Notable results:

- **Best classes**: T, U (F1-score = 1.0)

- **Challenging classes**: Q, Z (F1-score = 0.62 and 0.66 respectively)
- **Macro Avg F1-score**: 0.8192

### 4.3 Model Complexity

The model has ~5.6M trainable parameters, making it efficient compared to heavyweight models like ResNet or VGGNet【6】.

---

## 5. Conclusion and Future Work

The specific field studied proves that a specialized CNN could really carry out the job of classifying ISL gestures appropriately using PyTorch Lightning. Its modular design enables rapid experimentation with the codebase. Additional work concerns real-time gesture detection and its combination with dynamic gestures, plus deployment on edge devices using TorchScript or ONNX【7】【8】.

---

## References


1. Molchanov, P., Gupta, S., Kim, K., & Kautz, J. (2016). Hand gesture recognition with 3D convolutional neural networks. *CVPR*.
2. Kumar, D., Aggarwal, R. K., & Sharma, J. (2020). Sign language recognition using MobileNetV2. *Procedia Computer Science*, 173, 181-190.
3. Singh, J., Rana, A., & Kaushal, A. (2022). CNN-based classification for ISL gestures. *Procedia Computer Science*, 193, 221–228.
4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *CVPR*.
5. Falcon, W. (2019). PyTorch Lightning. GitHub repository. https://github.com/Lightning-AI/lightning
6. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
7. Hegde, R., & Patil, A. (2019). Sign language recognition on mobile platforms. *IEEE ICSC*.
8. Zhang, X., Yang, L., & Cai, J. (2019). Lightweight CNNs for real-time inference. *NeurIPS Workshop on Efficient ML*.
9. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *NIPS*.
10. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *CVPR*.


---