American Sign Language (ASL) Detection – Project Report

1. Introduction

This project focuses on building a computer vision system that can recognize **American Sign Language (ASL)** hand gestures. The goal was to classify 29 different signs (A–Z, SPACE, DELETE, NOTHING) using a deep learning model and deploy it through a simple Streamlit web app.

2. Dataset

The dataset used is the **ASL Alphabet Dataset from Kaggle**, which contains:

- 29 classes
- Around **3,000 training images per class**
- A separate folder for test images

The images cover a wide variety of hand positions, backgrounds, and lighting conditions, which helps the model learn more robust features.

3. Methodology

a. Preprocessing

- All images were resized to **64×64** for faster CPU training.
- Pixel values were normalized to the range **0–1**.
- The dataset was split using TensorFlow's validation_split feature (80% training, 20% validation).

b. Model Selection

To keep the project lightweight and workable on a normal CPU, I used **MobileNetV2** (pre-trained on ImageNet) with frozen base layers.
A custom classification head was added on top with:

- Global Average Pooling

- Dropout layer
- Dense output layer (29 units)

This allowed efficient transfer learning without needing heavy GPU resources.

c. Training

The model was trained for **5 epochs** due to system limitations.
Optimizer: **Adam**
Loss: **Categorical Crossentropy**

4. Results

- **Training Accuracy:** ~91%
- **Validation Accuracy:** ~64%

There is a noticeable gap between training and validation accuracy. This indicates **early overfitting**, which is expected because the model was trained for a limited number of epochs on a CPU.
With more time or GPU training, the validation accuracy is expected to improve significantly (typically 85–95% for this dataset).

5. Overfitting Observation

During training, the model performed extremely well on the training images but struggled a bit with unseen data.
This happens because:

- The dataset is large (around 87,000 images total).
- The model had limited time to learn generalizable features.
- Training was done on a standard laptop CPU.

Despite this, the model still performs decently as a **prototype** and correctly predicts many ASL test images.

6. Deployment

A simple **Streamlit app** (app.py) was built where users can upload an image, and the system displays the predicted ASL letter.
This makes the project interactive and easy to demonstrate.

7. Conclusion

This project successfully demonstrates a complete pipeline for ASL sign detection—from dataset preprocessing and model training to deployment through a web interface.
Even with restricted hardware, the model achieved promising results and proves the feasibility of real-time ASL recognition.
With further training on GPU and fine-tuning more layers of MobileNetV2, performance can be improved significantly.

8. Future Improvements

- Train for more epochs using GPU
- Add stronger data augmentation
- Fine-tune deeper layers of MobileNetV2
- Implement real-time webcam-based detection