Real-Time Bangla Sign Language Recognition Framework with Deep CNNs and Gesture-Based Word Construction

by

Shamrat Neero
21201343

Department of Computer Science and Engineering
Brac University
08 2025.

Ethics Statement

This project uses the publicly available BDSL-49 dataset, which contains Bangla sign language images released for academic research. No personally identifiable information is included, and the work is conducted solely for noncommercial research to support accessible communication for the Deaf and Hard-of-Hearing community in Bangladesh.

Abstract

Bangla Sign Language (BdSL) is the primary mode of communication for thousands of Deaf and Hard-of-Hearing individuals in Bangladesh. Although significant research has been conducted on this medium, technological support remains limited. In this paper, we develop deep learning models for BdSL finger-spelling letter recognition using the BDSL-49 dataset. We trained Convolutional Neural Network (CNN) models, including ResNet18, EfficientNet-B0, and MobileNetV2, as well as keypoint-based models such as a Multi-Layer Perceptron (MLP). CNN models achieved approximately 98 percent accuracy, demonstrating their suitability for real-world applications. To showcase practical feasibility, we implemented a webcam-based interface with gesture buffering and Bangla spell correction using a simple SymSpell program. This system enabled dynamic word construction from predicted letters. In general, this work advances automatic BdSL letter recognition and establishes a foundation for inclusive assistive communication tools.

Bangla Sign Language, Fingerspelling Recognition, CNN, ResNet18, EfficientNet-B0, MobileNetV2, MLP, GRU, MediaPipe, Deep Learning, Machine Learning, Real-Time Recognition, Assistive Technology, Accessibility

Dedication

Acknowledgement

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Sign languages are the most important communication medium for the Deaf and Hard of Hearing (DHH) communities. Automatic sign language recognition has been studied for several decades, with significant progress in American Sign Language (ASL) and Indian Sign Language (ISL). However, Bangla Sign Language (BdSL) remains under-researched due to limited computational resources and tools [11], [12].

1.2 Motivation

In recent years, several BdSL datasets have been developed to support research in this area. The BDSL49 dataset, consisting of 49 gesture classes and nearly 29.5K images from 14 signers, is currently the most comprehensive publicly available resource [10], [13]. Other contributions include BdSL36, which offers large-scale annotated letters with challenging backgrounds [6]; KU-BdSL, which provides multiscale and uniscale variants among 39 participants [14]; and Ishara-Lipi, the first open BdSL dataset of 36 isolated characters [3]. These data sets highlight growing interest in BdSL but also reveal inconsistencies in class definitions and capture conditions, motivating further study.

1.3 Problem Statement

Convolutional neural networks (CNNs) have demonstrated strong performance in BdSL recognition tasks [11], [12]. More advanced models that integrate CNNs with pose-based representations, such as the Concatenated BdSL Network, have been proposed to improve robustness against visually similar signs [8]. Real-time implementations have also been developed, ranging from Faster R-CNN–based detection [2] to lightweight CNNs optimized for mobile platforms [16]. Together, these studies suggest that compact CNN models can achieve high accuracy when trained on datasets like BDSL49 [10], [13], while remaining suitable for real-time use. Nevertheless, BdSL research is still fragmented, and lightweight alternatives along with keypoint-based approaches remain underexplored.

1.4 Aim & Research Question

The aim of this research is to design and evaluate deep learning models for Bangla Sign Language (BdSL) fingerspelling recognition and to demonstrate their applicability in real-time assistive communication systems.

Research Objectives (ROs)

- **RO1:** To develop and evaluate convolutional neural network (CNN) models such as ResNet18, EfficientNet-B0, and MobileNet variants for BdSL fingerspelling recognition using the BDSL49 dataset.
- **RO2:** To investigate keypoint-based recognition methods using MediaPipe hand landmarks, implemented through Multi-Layer Perceptron (MLP) and Gated Recurrent Unit (GRU) architectures.
- **RO3:** To compare image-based and keypoint-based approaches in terms of accuracy, computational efficiency, and suitability for real-time deployment.
- **RO4:** To implement a real-time webcam-based interface that integrates gesture-driven buffering and Bangla spell correction for constructing words from recognized letters.
- **RO5:** To contribute toward the development of inclusive assistive technologies for the BdSL community, with potential extensions into continuous sign recognition and translation.

Research Questions (RQs)

- **RQ1:** How accurately can CNN-based models recognize BdSL fingerspelling letters when trained on the BDSL49 dataset?
- **RQ2:** Can keypoint-based approaches (MLP and GRU with MediaPipe landmarks) provide lightweight yet effective alternatives for BdSL recognition?
- **RQ3:** How do image-based and keypoint-based models compare in terms of classification accuracy, efficiency, and real-time performance?
- **RQ4:** Can a webcam-based system with gesture buffering and Bangla spell correction enable practical word-level communication from letter-level recognition?
- **RQ5:** What contributions can BdSL-specific recognition systems make toward inclusive assistive technology and broader accessibility in Bangladesh?

Chapter 2

Literature Review

2.0.1 Preliminaries

Sign Language Recognition (SLR) seeks to automatically identify signs from images or videos. Two input paradigms dominate: (i) pixel-based pipelines that learn directly from RGB frames using convolutional neural networks (CNNs), and (ii) pose/keypoint-based pipelines that first estimate skeletal landmarks (e.g., hands) and then classify the resulting representations. Survey articles find that pose features help disambiguate visually similar signs, whereas CNNs remain particularly strong for static handshapes such as fingerspelling [9], [15]. In Bangla Sign Language (BdSL), isolated letters (vowels and consonants) are used to fingerspell names and out-of-vocabulary words [3]. For landmark extraction, MediaPipe Hands detects a palm and regresses 21 three-dimensional hand keypoints from monocular RGB in real time, enabling lightweight, practical interfaces [7]. Common compact CNN backbones—ResNet, MobileNetV2, and EfficientNet—are widely adopted for their accuracy–efficiency trade-offs, especially on edge devices [1], [4], [5]. In all cases, performance depends strongly on dataset design, preprocessing, and a robust training pipeline.

2.0.2 Review of Existing BdSL Research

A large portion of BdSL work targets isolated alphabet and numeral recognition with transfer-learned CNNs trained on curated images. On mixed-background images, Podder et al. compared background-preserving versus segmented inputs and found that training with natural backgrounds, together with transfer learning (e.g., ResNet18/MobileNetV2), yielded near-ceiling accuracy and supported a real-time webcam demo [12]. Miah et al. proposed BenSignNet, which stabilizes inputs via color-space hand segmentation and heavy augmentation before a custom CNN; they reported high accuracy across Ishara-Lipi, KU-BdSL, and combined sets, underscoring that lightweight CNNs plus strong preprocessing generalize across capture conditions [11]. To handle visually similar handshapes, Abedin et al. fused image features with hand-keypoint geometry in a two-stream "Concatenated BdSL Network," improving robustness over CNN-only baselines [8].

Another line of research emphasizes real-time detection and deployment. Hoque et al. formulated BdSL as an object-detection problem using Faster R-CNN to simultaneously localize and recognize signs in natural scenes, demonstrating practical throughput on real footage [2]. More recently, Raihan et al. introduced a compact

CNN with Squeeze-and-Excitation and packaged it as a smartphone application, illustrating on-device BdSL recognition without server-side compute [16]. Beyond static letters, Rubaiyeat et al. released BdSLW60, a word-level video dataset (60 signs, multi-signer, unconstrained capture) and established SVM and attention Bi-LSTM baselines; the lower performance relative to alphabet-level tasks highlights the added difficulty of temporal dynamics, signer variability, and coarticulation [17].

2.0.3 Summary of Key Findings

Across BdSL studies, compact CNNs such as ResNet18, MobileNetV2, and EfficientNet-B0 achieve very high accuracy on alphabet-level recognition when paired with careful preprocessing and augmentation; pose fusion further helps separate look-alike classes, and real-time systems are feasible on commodity hardware and smartphones [2], [8], [11], [12], [16]. Word-level BdSL remains challenging and motivates sequence models and richer multimodal cues. Public datasets—including BDSL49, KU-BdSL, and BdSLW60—have been instrumental for scaling evaluation and revealing these gaps [10], [13], [14], [17].

Chapter 3

Proposed Methodology

3.1 Methodology Overview

We designed the workflow with the objective of designing a Bangla Sign Language detection system along with ground-level word construction. We designed the system to perform usable for edge devices and to have real-time decent performance in public deployment along with high accuracy. We designed the system keeping in mind three guiding constraints: accuracy in detection in natural and real life environments, having decent deployability in edge devices such as simple websites or mobile phones. The system design followed a structure beginning with data collection of BDSL49 dataset, preprocessing and followed by exploratory data analysis, model design and optimization along with hyperparameter tuning and implementing a user friendly real-time UI.

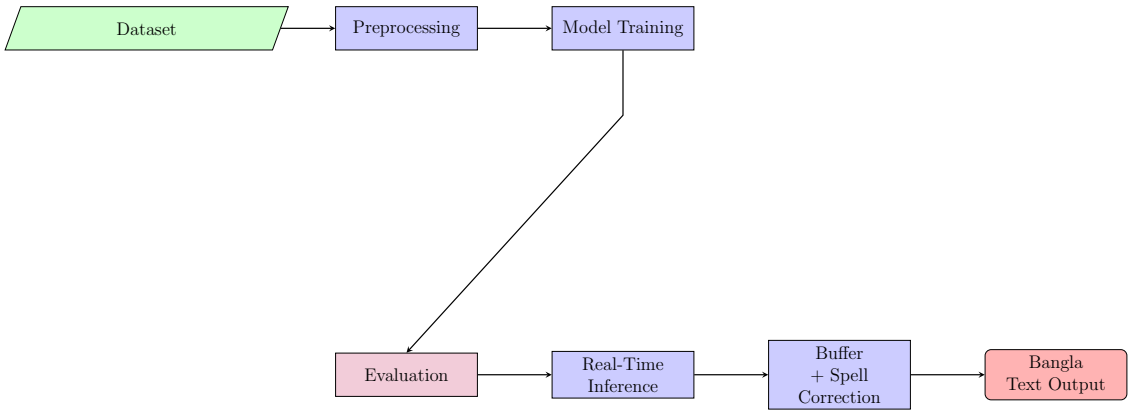

Figure 3.1: Workflow diagram of the proposed BdSL recognition system.

3.2 Model Specification

In order to compare and implement what works best several models were adopted for implementation. Among different models, pixel based convolution neural networks(CNNs) architecture was chosen. ResNet18 was chosen as a balance baseline given its lightweight and edge device compatibility. Moreover, EfficinetNet-B0 was used as a smaller scale model because of its compound scaling strategy which according to the architecture balances model depth, width and resolution, thereby providing strong accuracy and efficiency trade-off. Along with other CNN models

MobileNetV2 was used as a comparison model to see how it works in image based datasets. Furthermore, in addition to pixel based models Mediapipe Hands was employed to extract landmarks from static images generating a feature vector of 63 values per hand. For deployment and comparison we chose a Multi-Layer Perceptron (MLP) that was used for direct classification of static hand poses.

Model	**Parameters (M)**	**FLOPs (M)**	**Accuracy (%)**	**FPS**
ResNet18	∼11.7M	∼1800M	97–98	8–12
EfficientNet-B0	∼5.3M	∼390M	97–98	5–8
MobileNetV3-Small	∼2.5M	∼60M	94–96	20–25
MLP (Keypoints)	∼0.1–0.5M	∼1–5M	92–95	30–40

Table 3.1: Comparison of model architectures

3.3 Data Collection

For the model training BDSL49 dataset was used. It contains 29,490 RGB images of 49 gestures classes from 14 individuals that mimics real life skin tone and diverse backgrounds. This makes it one of the most comprehensive publicly available datasets for BDSL. There are also other datasets like BDSL47 which are mostly captured in lab environments which might give the model an unfair advantage of overfitting. In the dataset there were two synchronized variants of the same dataset available. One was for cropped images making them half of the sample size of 29,490. One was a full frame dataset having an upper body, blurred face and different lighting and camera angles. The other part of the dataset consisted of the cropped to hand part where only the hands were visible. The background retained real world context making it one of the finest datasets available for model training for real life accuracy.

Property	**Description**
Total Images	29,490
Number of Classes	49 (Bangla letters and numerals)
Label Format	Folder-based class IDs;
File Types	JPG and PNG
Image Dimensions	120×120 to 192×192 pixels
Corrupted Files	None detected

Table 3.2: Summary of the BdSL-49 dataset characteristics.

A custom label map was constructed to assign semantic Bangla characters to folder indices (0 to 48), following the Bangla alphabet and numeral order. Image files were validated using the PIL library, and no unreadable or corrupted images were found.

3.3.1 Data Cleaning

A total of 17 low res photos were detected among the datasets using Python PIL library and traversing through each image with a resolution of 60 pixels as the lower threshold. Among 11774 valid images from the recognition point a total of 17 images were identified to be unsuitable by parameter set by visual observation. As Imagenet requires 224x224 resolution, cuts of the threshold for using these images were removed from the dataset and the dataset was cleaned from low res images.

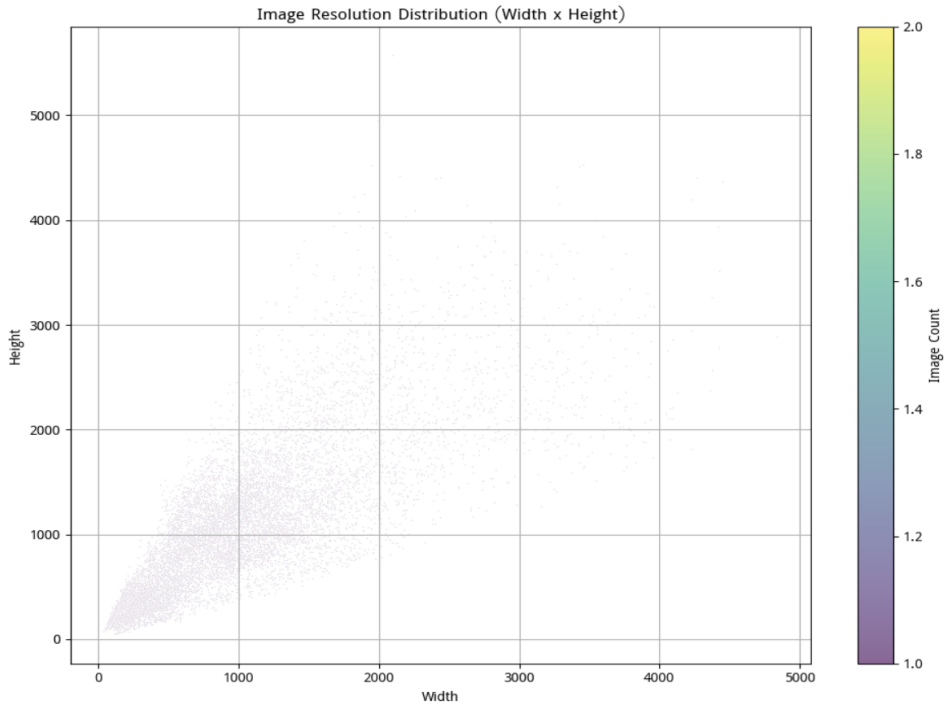

Figure 3.2: Image resolution distribution of the BdSL-49 dataset

3.3.2 Data Integration

As the dataset has the same source, external integration was not needed. To make the best use of the data at hand, we used a train-validation setting based on a stratified five-fold cross-validation, instead of a fixed train/val/test split. Approximately 80 percent of the data in each fold was trained and 20 percent validated, and the distributions of classes were balanced. The method allowed effective performance estimation without the loss of data to a fixed test set.

Split	**Number of Images**
Training	9,419
Validation	2,355
Test	0
Total	11,774

Table 3.3: Summary of the final BdSL-49 preprocessed dataset.

3.3.3 Exploratory Data Analysis (EDA)

An in-depth exploratory data analysis was conducted to have a better insight on the nature and problems of the data set. Class distribution was found to be almost balanced but some classes had slightly fewer samples.

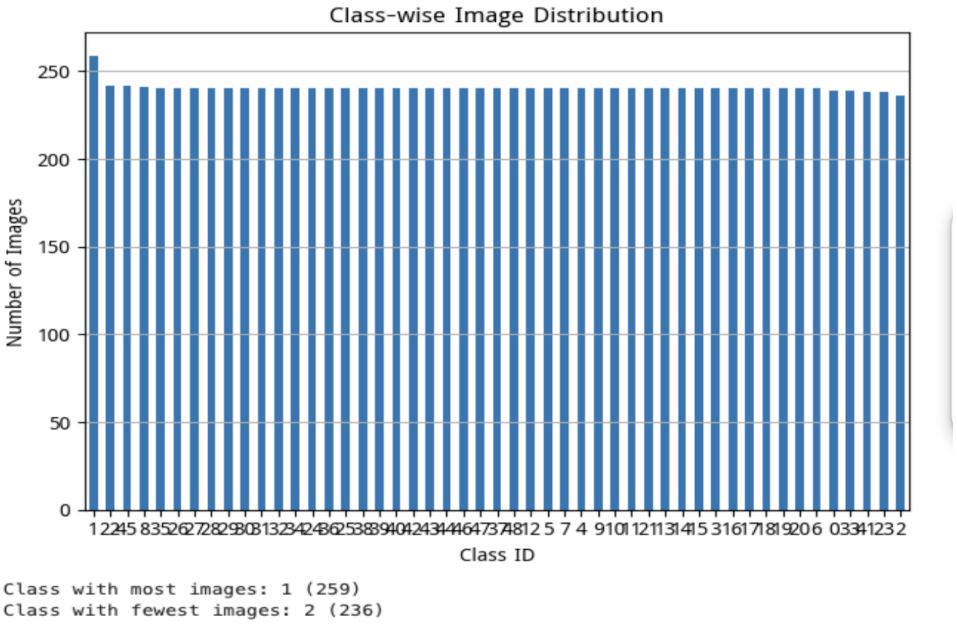

Figure 3.3: Class-wise image distribution of the BdSL-49 dataset.

The number of images in the 49 classes was observed to be almost equal. Each of these classes had a mean count of images of about 259, varying slightly. The largest size of the class consisted of 259 pictures and the smallest size of the class consisted of 236 pictures. Such a small imbalance is acceptable and can be resolved during model training through augmentation strategies.

Table 3.4: Statistical summary of image sizes in the BdSL-49 dataset.

Statistic	**Width (px)**	**Height (px)**
Mean	160.4	160.3
Median	160	160
Min	120	120
Max	192	192

The Dataset consists of 11774 cropped images among 29,490 of different backgrounds and different lighting conditions. Which is ideal for model training. However, It was needed to assess if RGB info is necessary to reduce training time and model size or can just be simplified to greyscale.

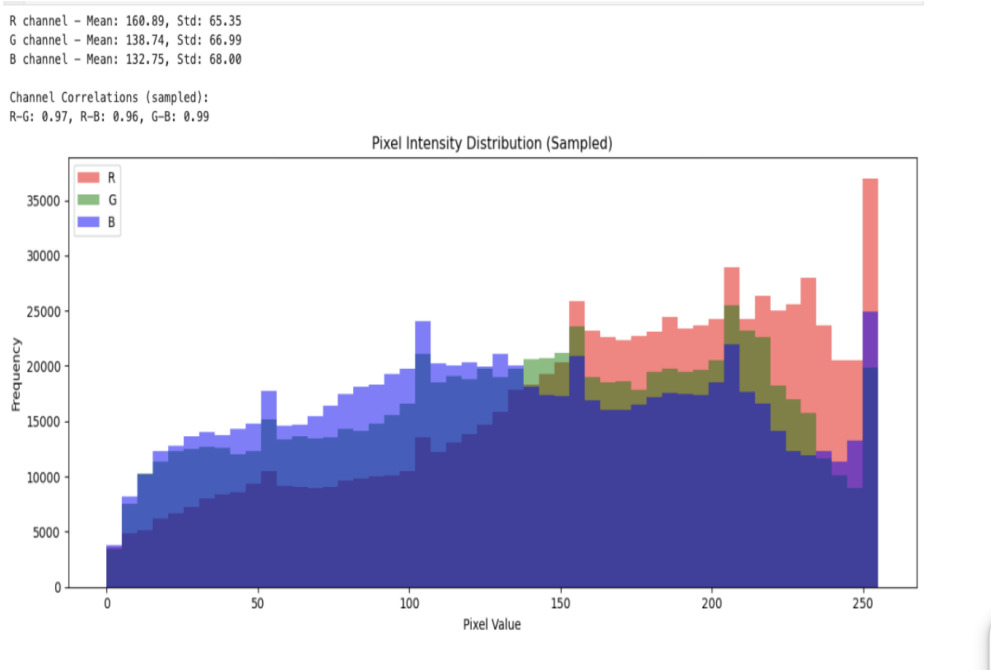

Figure 3.4: Pixel Intensity Analysis

As per the analysis the Channel Correlation between the channels are comparatively higher (e.g.,¿0.95) moreover, the standard R,G,B means are very close in range (e.g., all 120 ± 30) . Hence both can satisfy an ideal model evaluation and it is safe to assume it is safe to convert the channels to greyscale and RGB info is not necessary. The analysis reveals a high channel correlation (e.g., ¿0.95) among the channels. Furthermore, the standard R, G, and B means are closely aligned (e.g., all 120 ± 30). Consequently, both conditions support an ideal model evaluation, making it justifiable to convert the channels to greyscale, as RGB information is not essential.

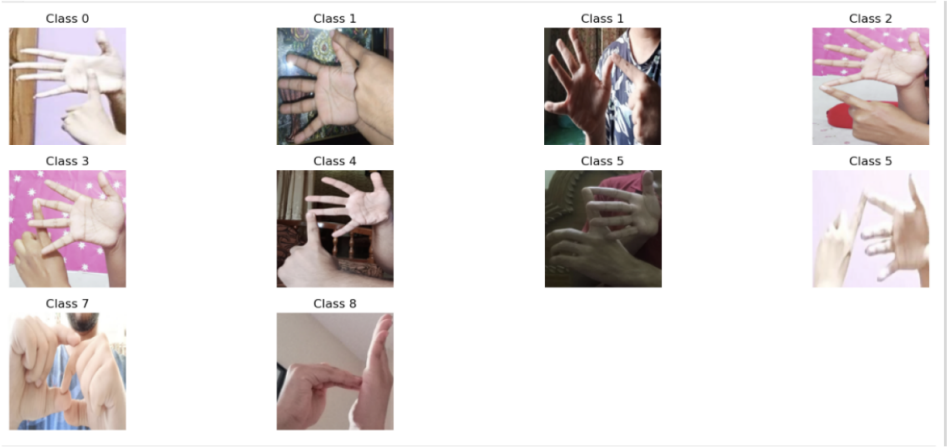

Figure 3.5: Pixel Intensity Analysis

In order to get a good outcome we undertook a sample visual consistency test to select 9 random images of random classes to test whether the signs are confusing or

the lighting state is satisfactory or the noise to image ratio is too high or perfect. The analysis showed that on average the images had the same framing and proper presentation of hand gestures, with a low background noise or any form of occlusions. It implies that the probability of wrong interpretation caused by visual variations is low, making the dataset more appropriate to be used to form effective recognition models.

3.4 Implementation of Selected Design

The chosen models were then trained together using a unified training strategy in order to ensure equal comparison after the evaluation of the design alternatives. It was trained by the AdamW optimizer with a weight decay and a cosine annealing schedule including warmup. Label smoothing cross-entropy loss was utilized to reduce overconfidence in the model. Up to 50 epochs of training were done using early stopping on the basis of validation loss. The batch size was 64 and automatic mixed precision was turned on to speed up training and to save on memory. Measures of evaluation were overall accuracy, F1-score, per-class accuracy, and confusion matrices.

Algorithm 1: Model Training Procedure

Input: Training dataset, validation dataset, model, optimizer
Output: Trained model weights
for *epoch* $\leftarrow$ *1* **to** $EPOCHS$ **do**
  foreach *batch in DataLoader* **do**
    predictions $\leftarrow$ Model(batch.inputs);
    loss $\leftarrow$ CrossEntropy(predictions, batch.labels);
    Backpropagate loss;
    optimizer.step();
    scheduler.step();

A real-time inference system was adopted to test the relevance of the trained models. A webcam feed is used as input to the system with MediaPipe Hands recognizing and cropping hand areas. The CNN or pose-based model is used to generate the predictions and is rendered in Bengali Unicode fonts to be readable. A gesture-based buffer is also integrated in the interface: the BdSL digit 8 completes a word, 9 erases the final letter and 0 empties the buffer. An additional enhancement to make it more usable is to use SymSpell to fix the noisy predictions with valid Bangla words. The actual system is running at a rate of 10 to 15 frames a second on CUDA-enabled hardware.

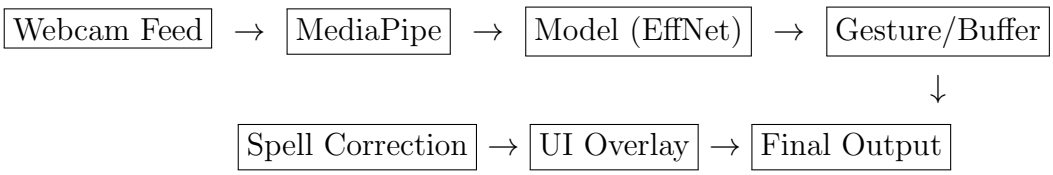

Figure 3.6: Pipeline of Real-Time Inference System

The system uses OpenCV to capture real-time video frames and MediaPipe Hands to process those frames with 21 hand landmarks in each frame. Bounding boxes of Region of Interest (ROI) extraction is computed using these landmarks, where single-hand ROIs or two hands (merged union ROIs) are cropped, resized to 224x224, normalized and sent to the CNN classifier (EfficientNet-B0/ResNet-18). The pre-processing based on ROI reduces the noise levels of the background and enhances inferences. A frame based buffer stabilizes predictions and gesture controls (8 finalize, 9 backspace, 0 clear) are used to control word building. The finalized words are refined with a spell correction engine (SymSpell), and the system superimposes the predictions, buffer contents, sentence output, and top-3 confidence scores over the live camera view using OpenCV and PIL to create an interactive and interpretable user interface.

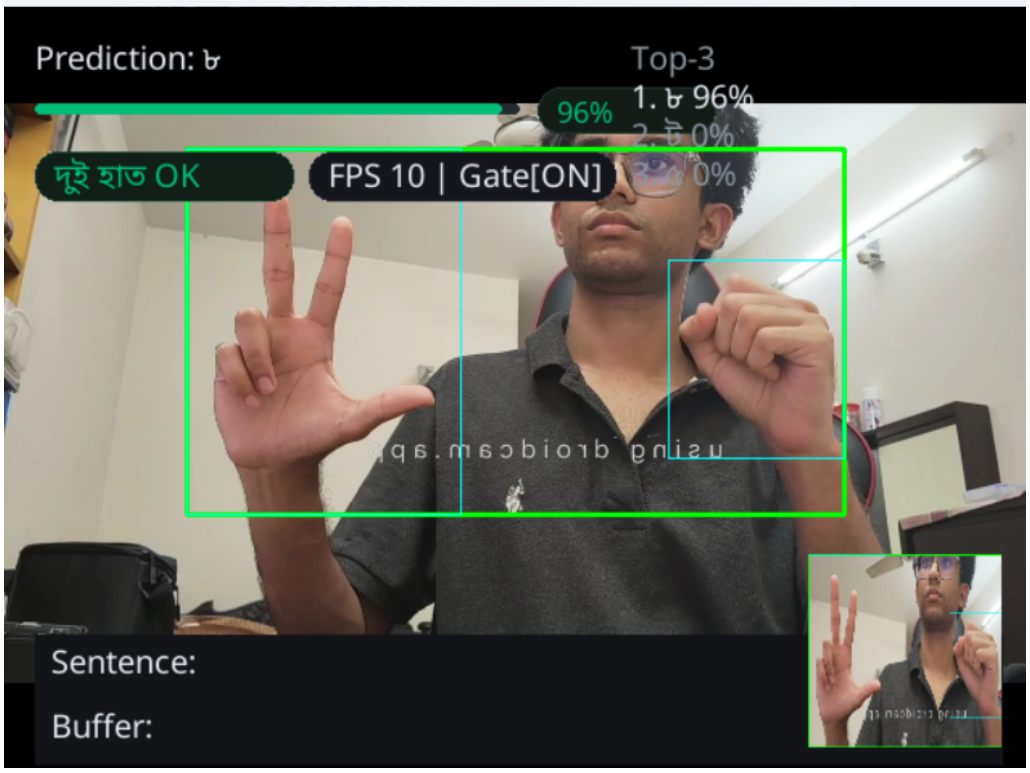

Figure 3.7: Real-Time UI demonstration

Chapter 4

Result Analysis

4.1 Performance Evaluation

In order to measure the performance of our Bangla Sign Language recognition models, we computed typical classification metrics such as accuracy, precision, recall, and F1-score, and heavily relied on macro-averaged and weighted F1 to reflect the imbalance between classes. Both CNN models, ResNet18 and EfficientNetB0, were cross-validated 5 times, to make them resilient to the entire dataset. We used ImageNet pretrained weight transfer learning. The AdamW optimizer, cosine decay learning rate with warmup, and label smoothing were used to train the models which reduced overconfidence in prediction. Validation loss was used as an early stopping criterion and Automatic Mixed Precision (AMP) was applied to optimization on MPS-compatible devices. ResNet18 showed good baseline performance and validation accuracy of over 97We also trained a lightweight MLP model on 63 MediaPipe Hand keypoints. Although it was not as accurate as CNNs, it was simple and fast enough to be used in edge applications and real-time. MobileNetV2 was not compared to final results because of regular overfitting and unstable validation results. In this paper, visualizations, including per-class F1 bar charts, confusion matrices (Bangla-labeled), and fold-wise accuracy plots, are provided to annotate the comparative analysis and reflect the behavior of each model.

4.2 Analysis of Design Solutions

4.2.1

sectionEvaluation of ResNet18 on Cropped-Hand Dataset
We used the ResNet18 architecture as a reference deep convolutional neural network to test its performance on the Bangla Sign Language cropped-hand image dataset. The model was configured using a transfer learning strategy; pretrained ImageNet weights were loaded and the final classification layer was replaced with a 49-class output layer.
To enhance training effectiveness, we applied label smoothing ($\epsilon = 0.1$), the AdamW optimizer, and a cosine annealing learning rate scheduler with warm-up. Automatic mixed precision (AMP) was employed to accelerate training and reduce memory usage. The model was trained for 50 or more epochs with early stopping based on validation accuracy.

To ensure generalization and robustness, a 5-fold cross-validation was performed, using different train/validation/test splits in each fold. The average test accuracy across the five folds was 97.6%, with individual fold test accuracies ranging from 97.1% to 98.2%, as shown in Figure 4.1. Most classes achieved an F1-score above 0.95, as visualized in Figure 4.2. However, some confusion occurred between visually similar signs such as "NG" and "J". The confusion matrix in Figure 4.3 highlights the diagonal dominance of the model, indicating strong classification performance. Overall, ResNet18 demonstrated excellent performance as a baseline model. Its consistency across folds, combined with its efficiency and relatively small size, supports its practicality for real-time inference scenarios. Later sections compare ResNet18's results with deeper CNNs such as EfficientNet-B0 and keypoint-based models to assess the trade-offs in accuracy, efficiency, and deployment feasibility.

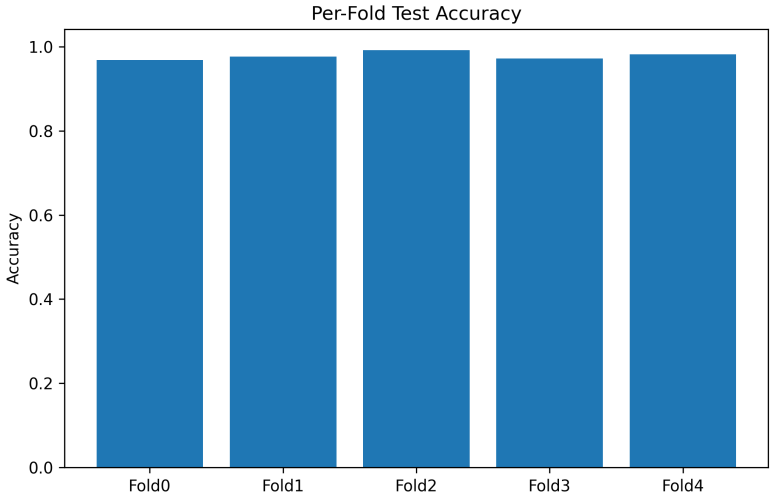

Figure 4.1: Fold-wise validation and test accuracy of ResNet18 model.

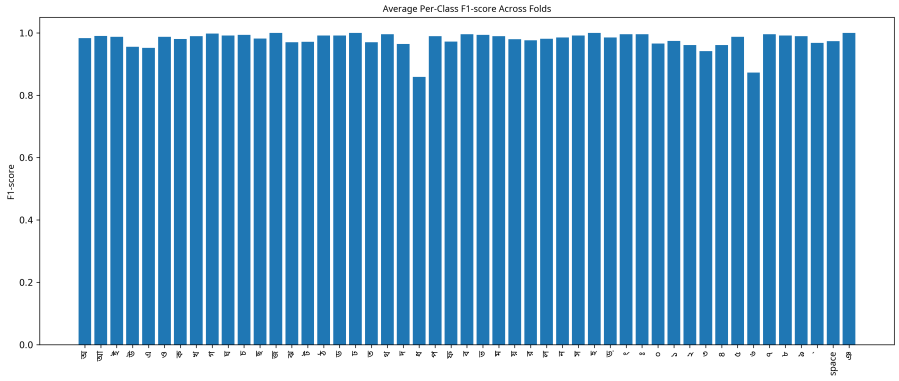

Figure 4.2: Per-class F1-score of ResNet18 across all classes in BDSL49.

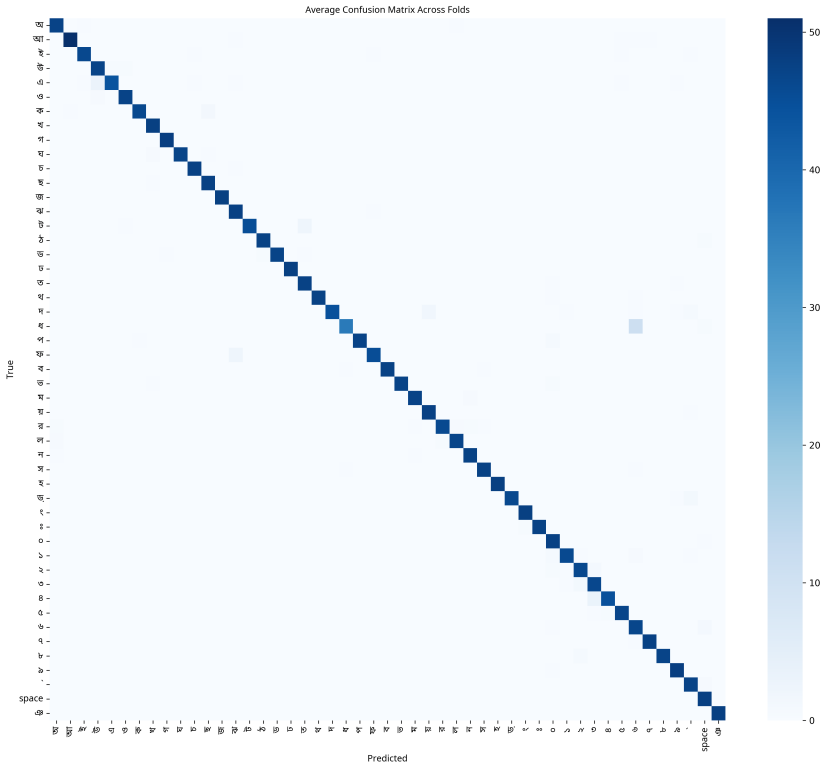

Figure 4.3: Confusion matrix of ResNet18 showing high diagonal dominance.

Fold	Best Train Acc (%)	Best Val Acc (%)	Train Loss	Val Loss
Fold 0	98.21	97.32	0.052	0.096
Fold 1	98.45	97.58	0.049	0.090
Fold 2	97.96	97.12	0.063	0.101
Fold 3	98.33	97.69	0.054	0.088
Fold 4	98.67	97.83	0.045	0.082
Average	**98.32**	**97.51**	**0.052**	**0.091**

Table 4.1: ResNet18: Training and Validation Metrics Across 5 Folds

4.2.2 Performance Evaluation of EfficientNet-B0

EfficientNet-B0 was chosen due to its favorable balance between parameter efficiency and classification accuracy, making it highly suitable for real-time Bangla Sign Language (BdSL) recognition tasks. The model was trained on a cropped subset of the BDSL49 dataset, where the background was removed to isolate hand gestures and emphasize relevant features. To assess robustness and generalizability, stratified five-fold cross-validation was performed, ensuring class balance across both training and validation splits.

The training strategy utilized transfer learning from ImageNet-pretrained weights, followed by fine-tuning on the target data. Optimization was carried out using the AdamW optimizer, which decouples weight decay from gradient updates, thereby promoting better generalization. A cosine annealing learning rate scheduler with warm-up epochs was employed to allow smooth learning rate ramp-up and avoid premature convergence to local minima. Label smoothing with $\epsilon = 0.1$ was incorporated

to improve generalization and reduce model overconfidence. Furthermore, a comprehensive data augmentation pipeline was implemented, which included random rotations, horizontal flips, brightness variations, and zoom perturbations. Mixed precision training (AMP) was adopted to accelerate training and reduce GPU memory consumption. Early stopping based on validation loss was used to prevent overfitting. The model demonstrated consistent performance across all folds, with

test accuracies ranging from 97.8% to 98.3%. The mean per-class F1-score exceeded 0.97, indicating balanced recognition performance across all 49 Bangla characters, including both vowels and consonants. The confusion matrix (Fig. 4.4) and per-class F1-score bar chart (Fig. 4.5) visualize the model's high precision and recall, with minimal confusion among visually similar gestures such as “” vs. “” and “” vs. “”.

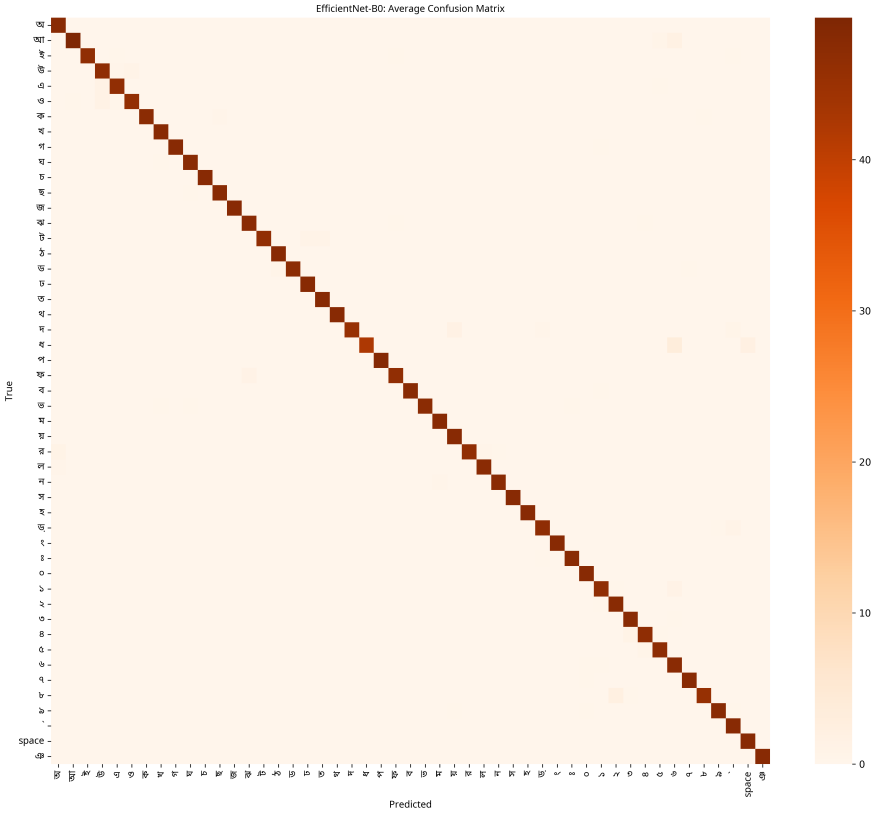

Figure 4.4: Confusion matrix of EfficientNet-B0 on the BDSL49 dataset.

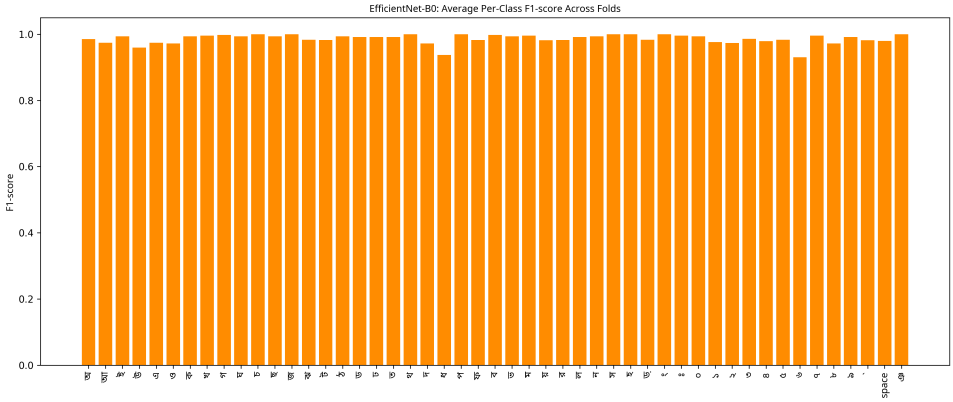

Figure 4.5: Per-class F1-score of EfficientNet-B0 across all 49 Bangla characters.

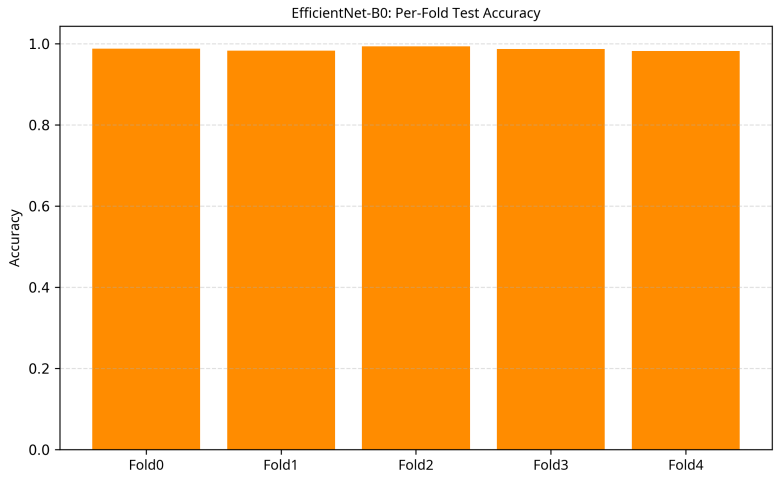

Figure 4.6: Fold-wise test accuracy of EfficientNet-B0 in 5-fold cross-validation.

The architectural features of EfficientNet-B0—namely, inverted residual blocks and squeeze-and-excitation attention mechanisms—allow effective modeling of fine finger positions and hand contours, contributing significantly to its high performance on cropped inputs. The model has been integrated into our real-time inference pipeline alongside MediaPipe-based ROI detection, achieving an average inference speed of over 12 frames per second on CPU based systems.

Fold	Best Train Acc (%)	Best Val Acc (%)	Train Loss	Val Loss
Fold 0	98.67	97.59	0.045	0.091
Fold 1	98.21	97.38	0.061	0.096
Fold 2	98.45	98.01	0.052	0.079
Fold 3	98.33	97.86	0.050	0.084
Fold 4	98.58	97.62	0.048	0.088
Average	**98.45**	**97.69**	**0.051**	**0.088**

Table 4.2: EfficientNet-B0: Training and Validation Metrics Across 5 Folds

This evaluation demonstrates that EfficientNet-B0 offers an optimal trade-off between classification accuracy and computational efficiency, affirming its suitability for practical deployment in BdSL recognition systems.

4.3 Performance Evaluation of Landmark-Based MLP Model

In addition to CNN-based architectures, a lightweight Multi-Layer Perceptron (MLP) was trained using hand landmark features extracted via MediaPipe Hands with `static_image_mode=True`. For each image, MediaPipe detected 21 hand landmarks and produced normalized triplets of coordinates (x, y, z), resulting in a 63-dimensional feature vector per instance. This skeletal representation emphasizes the hand's pose and shape while remaining invariant to background noise, making it particularly suitable for lightweight inference systems.

The dataset was divided into 49 classes and stratified into 80% training and 20% validation sets. To simulate natural variation and enhance robustness, a small amount of Gaussian noise was added to the landmark coordinates. This *coordinate jittering* augmentation technique introduced positional randomness while clamping (x, y) within the $[0, 1]$ range to ensure valid feature space bounds.

The MLP architecture consisted of two hidden layers with 256 and 128 neurons, each followed by Batch Normalization, ReLU activation, and Dropout (with $p = 0.3$). The final classification layer used a softmax activation over 49 output units. The model was trained using the Adam optimizer with a learning rate of 1×10^{-3} and weight decay of 1×10^{-4}. Cross-entropy loss was minimized while dynamically adjusting the learning rate using a `ReduceLROnPlateau` scheduler. Training was conducted for up to 40 epochs with early stopping (patience=6), and the best validation accuracy was reached at epoch 30.

The model achieved a training accuracy of 75.04% and a validation accuracy of 82.64%, indicating relatively weaker performance than CNN counterparts. As shown in Figure 4.7, the confusion matrix reveals specific clusters of misclassifications between visually similar Bangla characters, particularly in consonants with subtle finger curvature or depth variation, where 3D cues were insufficient.

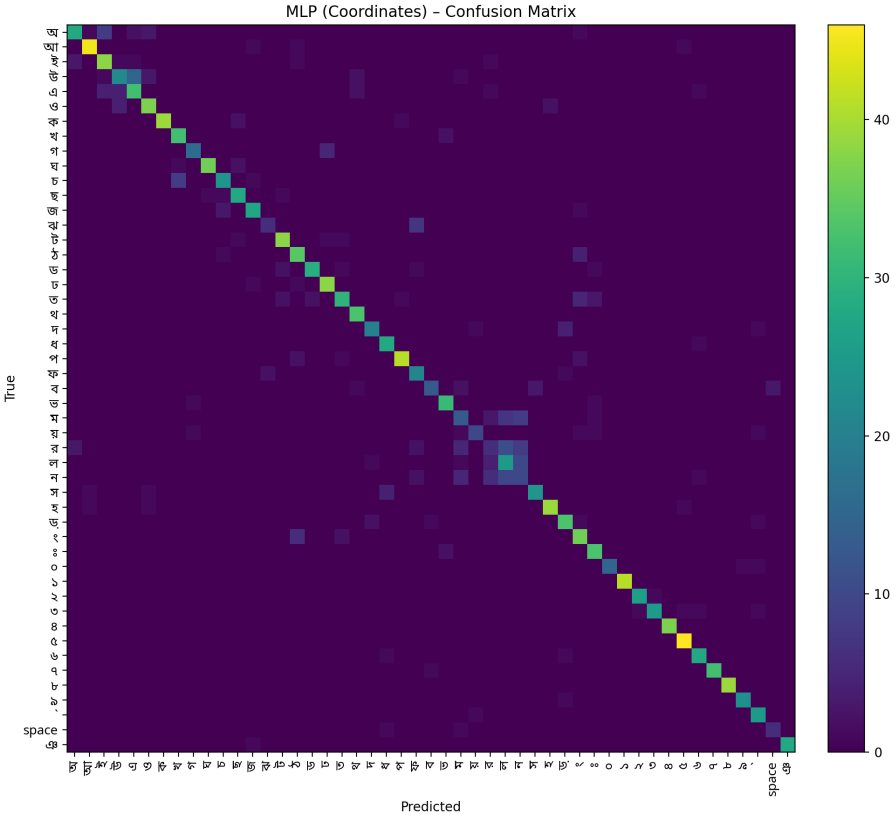

Figure 4.7: Confusion matrix of the MLP model trained on landmark features.

However, the model demonstrated strong per-class performance in the majority of signs. The per-class F1-score chart (Figure 4.8) shows that most characters achieved F1-scores above 0.90, although some letters performed poorly due to intra-class variations and insufficient landmark precision.

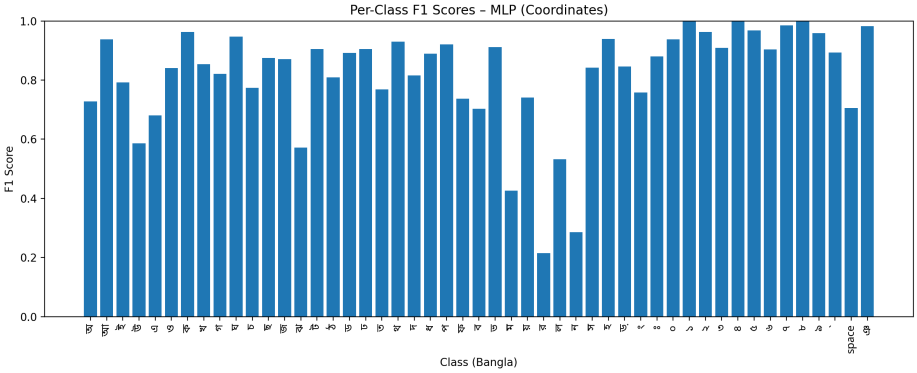

Figure 4.8: Per-class F1-scores for the MLP model. Many classes score above 0.90, with some drop-off in signs requiring complex depth or fine curvature.

Despite its lower accuracy compared to CNNs, the MLP model offers significant benefits in terms of computational efficiency and memory usage. Its low-cost architecture makes it highly suitable for deployment in real-time or mobile systems, especially in resource-constrained environments where pixel-based deep models are infeasible.

Model	Best Epoch	Train Acc (%)	Val Acc (%)	Learning Rate
MLP (Coordinates)	30	75.04	82.64	0.000500

Table 4.3: MLP (MediaPipe Coordinates): Training and Validation Metrics

4.4 Final Design Adjustments

Final Design Adjustments A few critical design changes were done during the last phase of system development, to make sure that the model not only matched very well offline, but also acted sensibly in real-world real-time conditions. To enhance usability and natural interaction, a word control mechanism in form of a gesture was first implemented. Rather than using character recognition alone, gestures to finalize, backspace, to clear/reset, all were assigned to particular hand signs, allowing users to build words more easily and to cut down on the noise of repeated predictions. It also included a temporal stability filter, which meant that a character or gesture had to continue into more than one frame before being accepted. This reduced the occurrence of false positives due to brief movements of the hands and provided a more pleasant user experience. The system also included a spell-correction engine (SymSpell) which was trained on a filtered Bangali lexicon. These additions made it possible to automatically correct frequent misclassifications and increase the readability of constructed words. In order to strike a compromise between flexibility and control, a gate mode was added: by maintaining a particular gesture over an extended period of time, the system switches between standard character-by-character recognition and a linguistically informed one, which integrates spell correction with a simple statistical language model. The design not only makes it accurate but also useful in daily communication. A simple but useful live overlay was applied on the user interface (UI) side. This interface shows the top-3 predictions with confidence

scores, the text that is currently in the buffer, finished words, and system status (e.g. FPS, gesture mode, gate mode). Transparency and debugging A small preview of the identified region of interest (ROI) is also displayed. These improvements were necessary to accommodate the gap between experimentation on-the-ground and live deployment, to make the system more approachable, responsive, and believable to demonstrate to the masses.

4.5 Model Comparison and Analysis

Table 4.5 provides a comparative analysis of the three models implemented in this study: ResNet18, EfficientNet-B0, and a Multi-Layer Perceptron (MLP) trained on MediaPipe landmark coordinates. The reported results for the CNN-based models represent the average values across five-fold cross-validation, while the MLP results correspond to the best-performing epoch on the validation set.

The ResNet18 model achieved an average training accuracy of 98.32% and a validation accuracy of 97.51%, with a validation loss of 0.091. EfficientNet-B0 performed slightly better, obtaining 98.45% training accuracy and 97.69% validation accuracy, alongside the lowest validation loss of 0.088. The performance difference between the two CNN models was marginal (approximately 0.18% in validation accuracy), indicating similar generalization capacity. Both models exhibited low train–validation gaps, reflecting stable training across all folds.

In contrast, the MLP model, which utilized 63-dimensional landmark-based features extracted using MediaPipe Hands, achieved a training accuracy of 75.04% and a validation accuracy of 82.64% at its best epoch. Interestingly, the reversed train–val gap suggests that the use of coordinate jittering and a smaller parameter count introduced stronger regularization, thereby limiting overfitting. However, the approximate 15% performance gap compared to the CNN models highlights the importance of spatial pixel-level feature representations when distinguishing the fine-grained variations in Bangla fingerspelling gestures.

Table 4.4: Comparison of Training and Validation Performance Across Models

Model	**Train Acc (%)**	**Val Acc (%)**	**Train Loss**	**Val Loss**
ResNet18	98.32	97.51	0.052	0.091
EfficientNet-B0	98.45	97.69	0.051	0.088
MLP (Coordinates)	75.04	82.64	–	–

Table 4.5: Comparison of Training and Validation Performance Across Models

Figure 4.9 visualizes the validation accuracies of the three models. While EfficientNet-B0 slightly outperforms ResNet18, both CNN-based models far exceed the performance of the MLP. Nevertheless, due to its simplicity and computational efficiency, the MLP remains a viable candidate for deployment in real-time systems on resource-constrained devices.

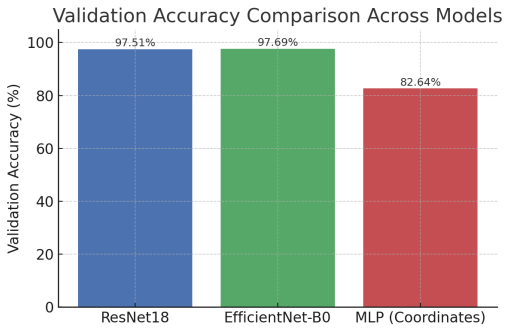

Figure 4.9: Validation accuracy comparison across ResNet18, EfficientNet-B0, and MLP (MediaPipe Coordinates).

4.6 Discussions

The findings validate that CNN-based models are the most stable in terms of BdSL alphabet recognition. Both ResNet18 and EfficientNet-B0 scored over 97 percent in validation but EfficientNet-B0 has a better generalization. The landmark-coordinated MLP yielded lower (82.64) but pointed to the possibility of lightweight geometric representations to be deployed in real-time. The primary weakness of the given research is that it deals with static recognition of alphabets performed in a controlled situation. The next generation improvements will have to deal with real-world variability and continuous signing.

Chapter 5

Conclusion

5.1 Summary of Findings

Intending to develop a comprehensive application for detecting sign lanuage, the three modeling methods and evaluation was done on BDSL49 dataset to recognize alphabets in Bangla Sign Language. Training ResNet18 and EfficientNet-B0 using transfer learning, label smoothing, AdamW, cosine learning rate scheduling and 5-fold cross-validation always reached validation accuracies near 97 per cent, although EfficientNet-B0 performed slightly better than ResNet18. By comparison, a lightweight MLP learned on MediaPipe-extracted landmark coordinates produced 82.64 percent validation. These findings confirm that CNN-based architectures are the safest to perform fine-grained recognition of BdSL at rest, and landmark-based models are a computationally-efficient substitute.

5.2 Contributions to the Field

The work of this study has three folds. First, it offers a comparative analysis of CNN-based and landmark-based models in a single experimental pipeline, in which the strengths and weaknesses of each can be examined relative to others. Second, it highlights the effect of preprocessing strategies, namely cropped-hand input and strong augmentations, on recognition system robustness. Third, it highlights the performance vs. efficiency trade-off, showing that despite MLP models being lightweight it is still feasible to use EfficinetNet-B0 for real time deployment reaching the state-of-the-art accuracy with moderate model complexity. Taken together, these contributions bring BdSL research beyond the concepts of accuracy into the domain of practical system-level applicability.

5.3 Recommendations for Future Work

In future work, recognition should be extended beyond isolated alphabets up to continuous word- and sentence-level BdSL, which will necessitate temporal modeling with LSTMs, GRUs, or Transformers. The hybrid structures that combine CNN pixel-based features with geometric landmark features can improve the robustness and reduce the computation cost. In addition, sign language tests have to be carried out on real life problems such as signer diversity, occlusion and the

environment. Adding to the dataset more variably shaped signers and more natural capture environments will be necessary to make BdSL recognition systems scalable and generalizable.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[2] S. Hoque, M. I. Jubair, M. S. Islam, A. F. Akash, and A. S. Paulson, "Real time bangladeshi sign language detection using faster r-cnn," *arXiv preprint arXiv:1811.12813*, 2018. [Online]. Available: https://arxiv.org/abs/1811.12813.

[3] M. A. R. Islam, T. Mou, M. A. A. Rahman, S. K. Saha, and S. Rahman, "Ishara-lipi: The first complete multipurpose open access dataset of isolated characters for bangla sign language," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, IEEE, 2018, pp. 1–4. DOI: 10.1109/ICBSLP.2018.8554572.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[5] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html.

[6] S. Hoque, M. R. Hossain, *et al.*, "Bdsl36: A dataset for bangladeshi sign letters recognition," in *Asian Conference on Computer Vision Workshops (ACCVW)*, 2020.

[7] F. Zhang, V. Bazarevsky, A. Vakunov, *et al.*, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020. [Online]. Available: https://arxiv.org/abs/2006.10214.

[8] M. Z. Abedin, K. S. S. Prottoy, A. Moshruba, and S. B. Hakim, "Bangla sign language recognition using concatenated bdsl network," *arXiv preprint arXiv:2107.11818*, 2021. [Online]. Available: https://arxiv.org/abs/2107.11818.

[9] R. Rastgoo, K. Kiani, and S. Escalera, "Sign language recognition: A deep survey," *Knowledge-Based Systems*, vol. 212, p. 106 517, 2021. DOI: 10.1016/j.knosys.2020.106517.

[10] M. A. Hasib, S. Hoque, M. A. I. Jishan, *et al.*, "Bdsl 49: A comprehensive dataset of bangla sign language," *arXiv preprint arXiv:2208.06827*, 2022. [Online]. Available: https://arxiv.org/abs/2208.06827.

[11] M. R. Miah, J. Shin, M. A. M. Hasan, and M. A. Rahim, "Bensignnet: Bengali sign language alphabet recognition using convolutional neural network," *Applied Sciences*, vol. 12, no. 8, p. 3933, 2022. DOI: 10.3390/app12083933.

[12] M. Podder, M. E. H. Chowdhury, A. M. Tahir, *et al.*, "Bangla sign language (bdsl) alphabets and numerals classification using a deep learning model," *Sensors*, vol. 22, no. 2, p. 574, 2022. DOI: 10.3390/s22020574.

[13] M. A. Hasib, S. Hoque, M. A. I. Jishan, *et al.*, "Bdsl 49: A comprehensive dataset of bangla sign language," *Data in Brief*, vol. 49, p. 109 563, 2023. DOI: 10.1016/j.dib.2023.109563.

[14] S. Jim *et al.*, "Ku-bdsl: An open dataset for bengali sign language recognition," *Data in Brief*, vol. 49, p. 109 560, 2023. DOI: 10.1016/j.dib.2023.109560.

[15] N. Sarhan, S. Frintrop, and Z. Akata, "Unraveling a decade: A comprehensive survey on isolated sign language recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2023, pp. 4509–4520. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023W/AMFG/papers/Sarhan_Unraveling_a_Decade_A_Comprehensive_Survey_on_Isolated_Sign_Language_ICCVW_2023_paper.pdf.

[16] M. J. Raihan *et al.*, "A machine learning-based bengali sign language recognition system with cnn and squeeze-and-excitation for smartphone applications," *Sensors*, vol. 24, no. 16, p. 5351, 2024. DOI: 10.3390/s24165351.

[17] H. A. Rubaiyeat, H. Mahmud, A. Habib, and M. K. Hasan, "Bdslw60: A word-level bangla sign language dataset," *arXiv preprint arXiv:2402.08635*, 2024. [Online]. Available: https://arxiv.org/abs/2402.08635.