BANGLA SIGN LANGUAGE RECOGNITION: DATASET INNOVATION AND METHODOLOGIES

By

1. Saad Ahmed ID: 200201008
2. Razia Sultana ID: 200201007

B.Sc. Engineering in Computer Science and Engineering

Department of Computer Science and Engineering

Faculty of Electrical and Computer Engineering

Bangladesh Army University of Science and Technology (BAUST)

JUNE 2024

Bangla Sign Language Recognition: Dataset Innovation and Methodologies

This thesis is submitted in partial fulfillment of the requirement for the degree of B.Sc. Engineering in Computer Science and Engineering

By

1. Saad Ahmed ID: 200201008

2. Razia Sultana ID: 200201007

Supervised by

Dr. S.M. Jahangir Alam

Professor and Head, Department of CSE

Co-Supervised by

Md. Khalid Syfullah

Lecturer, Department of CSE

Department of Computer Science and Engineering

Bangladesh Army University of Science and Technology (BAUST)

Saidpur Cantonment, Nilphamari, Bangladesh

The thesis titled **"Bangla Sign Language Recognition: Dataset Innovation and Methodologies"** submitted by Saad Ahmed (ID:200201008) and Razia Sultana (ID:200201007), session 2020-2021 has been presented on 24/06/2024 and accepted as satisfactory in fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering (CSE) as B.Sc. Engineering to be awarded by the Bangladesh Army University of Science and Technology (BAUST).

Board of Examiners

1. _____ Supervisor

   Dr. S.M. Jahangir Alam
   Professor and Head
   Department of Computer Science and Engineering (CSE)
   Bangladesh Army University of Science and Technology (BAUST)

2. _____ Co-Supervisor

   Md. Khalid Syfullah
   Lecturer
   Department of Computer Science and Engineering (CSE)
   Bangladesh Army University of Science and Technology (BAUST)

3. _____ Member

   Dr. Engr. Mohammed Sowket Ali
   Associate Professor
   Department of Computer Science and Engineering (CSE)
   Bangladesh Army University of Science and Technology (BAUST)

4. _____ Member

Hasan Muhammad Kafi

Assistant Professor

Department of Computer Science and Engineering (CSE)

Bangladesh Army University of Science and Technology (BAUST)

5. _____ Member

Md. Zahid Hassan

Lecturer

Department of Computer Science and Engineering (CSE)

Bangladesh Army University of Science and Technology (BAUST)

6. _____ Member

Md. Mahadi Hasan

Lecturer

Department of Computer Science and Engineering (CSE)

Bangladesh Army University of Science and Technology (BAUST)

_____

Dr. S.M. Jahangir Alam

Professor and Head

Department of Computer Science and Engineering (CSE)

Bangladesh Army University of Science and Technology (BAUST)

CANDIDATE'S DECLARATION

It is hereby declared that the contents of this thesis are original and any part of it has not been submitted elsewhere for the award of any degree or diploma.

1.	Saad Ahmed	200201008	
2.	Razia Sultana	200201007	

ACKNOWLEDGMENTS

First and foremost, we are deeply grateful to Almighty Allah for granting us the strength and perseverance to complete this thesis successfully. We extend our sincerest thanks and heartfelt gratitude to our honorable thesis supervisor, Dr. S.M. Jahangir Alam, Professor and Head of the Department of Computer Science & Engineering (CSE) at Bangladesh Army University of Science & Technology (BAUST), Saidpur. His inspiration and unwavering support have been instrumental in the completion of our thesis.

We like to express our special thanks to the Head of the Department of Computer Science & Engineering (CSE), Dr. S.M. Jahangir Alam for his valuable suggestions, positive advice, encouragement, and sincere guidance throughout our thesis work. Additionally, we convey our gratitude to all the respected teachers of the Department of CSE, as well as those from other departments, for their support and contributions.

Our sincere thanks go to the respected Honorable Vice Chancellor, the Chief of Army Staff and board of trustees, as well as all members and academic staff who have led this institution and contributed to our bright future.

We would also like to thank all our friends and the staff of the department for their valuable suggestions and assistance. Finally, we extend our deepest appreciation to our parents and families for their unwavering love and support throughout our studies.

ABSTRACT

Bangla Sign Language (BdSL) is essential for communication among people with hearing impairments in Bangladesh. In order to address the need for improved BdSL gesture detection, this paper introduces the RSBdSL38 dataset. To ensure variety and usefulness, 11,864 gestures from 46 individuals in various difficult situations are included in the dataset. With its improvements over earlier datasets, RSBdSL38 represents a major advancement in BdSL recognition technology. We tested the dataset's ability to aid in the understanding of BdSL gestures using sophisticated machine learning algorithms. We examined important aspects such as accuracy and performance in various scenarios and contrasted our findings with available datasets to demonstrate the power and use of RSBdSL38. The foundation for developing more inclusive and accurate BdSL recognition systems is laid by this research, which will facilitate everyday communication for BdSL users.

Keywords: Gesture Recognition, Pretrained Models, Ensemble Techniques, Robustness and Comparative Analysis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Background

Individuals who are deaf or hard of hearing rely on sign language as their natural and primary way of communication. Humans use signs, gestures, and facial expressions to convey meaning naturally from birth, underscoring sign language's innate role as a main mode of contact. While sign language promotes seamless communication among the deaf and hard-of-hearing community, it presents obstacles for those who do not understand its complexities.

For people who are unable to talk or hear, sign language becomes the sole means of successful communication. Its universal accessibility makes it essential for people with speech and hearing difficulties, providing them with a critical lifeline for expressing their thoughts, feelings, and ideas. However, bridging communication between sign language users and those unfamiliar with its lexicon and grammar is a substantial challenge. Because there are so many different languages in the globe, sign language varies a lot from nation to nation and has unique linguistic characteristics. Though regional variants like Bengali Sign Language (BdSL) serve particular linguistic communities, American Sign Language (ASL) is an internationally recognized form of sign language [11].

With the increasing use of sign language and technological improvements, there is a great chance to improve accessibility and understanding. Real-time interpretation is made possible by systems that employ computer vision and machine learning techniques. This helps to close the communication gap between people who use sign language and others. This introduces collection data specific to the grammatical nuances and cultural background of Bengali Sign Language (BdSL) (Figure:1.1) which shows the need for comprehensive datasets for this language [12].

Figure 1.1: Bangla Sign Language (BdSL) gestures [1].

1.2 Problem Statement

Bridging the gap between natural language and sign language is a major difficulty in the field of continuous sign language recognition. Fluent continuous sign language translation cannot be achieved without accurate word-level recognition, which is a prerequisite for sign sentences, which are composed of discrete sign words and sometimes fingerspelling for proper nouns [13].

There are two major problems that dominate the field of sign language studies. First, a basic challenge is the lack of extensive datasets customized to the complexities of sign language. Secondly, a persistent difficulty is the creation of machine learning and deep learning models that can capture all aspects of articulated sign context. Recentrly researcher has expressed need of large scale word-level sign language recognition which involves both need for proper datasets, machine learning and deep learning models [14]. A considerable segment of the populace in Bangladesh utilizes Bangladeshi Sign Language (BdSL) for routine communication [15]. A digital communication tool that closes the gap between signers and non-signers is therefore necessary. But even though computers are getting better at identifying hands in pictures [16, 17], there isn't much focus

on real-time, speedy recognition of individual sign letters, especially when it comes to Bangladeshi Sign Language. This difficulty is made much more difficult by the dearth of relevant datasets designed specifically to aid in this undertaking.

1.3 Objectives

The primary objective of this research is the development of a comprehensive dataset of Bangla Sign Language (BdSL) gestures. Recognizing the importance of data in training and testing machine learning and deep learning models, significant efforts were made to curate a dataset of 12,000 photos. The study's foundation is built on this dataset, which captures the subtle motions and expressions inherent in BdSL communication. The dataset's goal is to provide a solid foundation for future model development and evaluation by including a wide range of indications and movements.

Building on the foundation built by the curated dataset, the next goal of this research is to deploy machine learning and deep learning models for Bengali sign language detection. These models use cutting-edge approaches and computational tools to accurately and efficiently recognize BdSL gestures from visual data. The study aims to use advanced algorithms and neural network architectures to harness the power of artificial intelligence in decoding the complexities of sign language communication.

The research focuses on optimizing and improving the created models to improve the accuracy and robustness of Bangla Sign Language (BdSL) detection after the model implementation and performance evaluation phases. This entails investigating cutting-edge techniques and iteratively fine-tuning model parameters in order to advance sign language recognition technology. The ultimate goal of the research is to apply its findings to real-world situations where they can help signers and non-signers communicate easily, promoting accessibility and inclusivity for people with hearing loss in a variety of contexts.

1.4 Social Impact

This study aims to improve the quality of life for people with hearing impairments by developing accurate sign language identification systems particular to Bangla Sign

Language (BdSL). The study's goal is to develop more understanding and acceptance in society by allowing for smooth interactions between signers and nonsigners. The study addresses common stigmas and misconceptions about hearing impairments by increasing communication between those who use sign language and those who don't, resulting in a more compassionate and welcoming community.

Furthermore, the implications of this research go beyond interpersonal interactions and include educational, healthcare, and employment settings. Improved communication accessibility is expected to result in more inclusive learning environments, better healthcare outcomes, and increased integration of people with hearing impairments into the workforce, increasing chances for economic inclusion. Finally, this study advocates for the rights and dignity of persons with hearing impairments, aiming to enable their full involvement in social, economic, and cultural arenas while also contributing to the creation of a more equitable and inclusive society.

1.5 Issues Encountered

During the course of this research, I encountered several challenges that posed significant obstacles to progress. One of the main obstacles was the lengthy data collecting procedure, which required traveling to several schools in Bangladesh that serve autistic children in order to acquire pictures from students who were deaf and silent. It was extremely difficult to communicate with these students since it took time, empathy, and specific methods to build trust and promote meaningful relationships. The study became more complex due to the necessity of substantial travel and logistical coordination due to the sheer magnitude of data collection. In addition, after obtaining the dataset, I ran into difficulties while putting several machine learning and deep learning algorithms into practice. Although these algorithms produced a multitude of results, such as graphs, matrices, and figures, it was difficult to choose the best approaches and analyze the results. It took careful navigation and iterative refinement to strike a balance between the practical issues of real-world application and the technical complexities of algorithm selection.

1.6 Organization of the Thesis

We started the book with a thorough examination of sign language as a natural form of communication, emphasizing its intrinsic value and how important it is for smooth communication, especially for those who are deaf or hard of hearing. We emphasized how crucial effective recognition systems are to the field of sign language technology and how they may help signers and non-signers communicate more effectively in a variety of real-world situations. As we go on to the next chapter, a thorough review of the body of research in the area was done, along with an analysis of the methods used and limitations identified in earlier attempts to perform sign language recognition study. The next chapter then describes the methodically developed approach and the finely detailed implementation of the suggested algorithm for sign language recognition. This involves applying state-of-the-art machine learning and deep learning algorithms, as well as creating a broad and varied dataset with a variety of sign language motions and expressions. In the domain of result analysis, we carefully examine the experimental results and offer insightful comparisons with current recognition methodologies. In the final section, we summarize the key findings from this study, explaining their importance for the development of sign language recognition technology and suggesting future directions for this important field of study. This research work is publicly available at GitHub repository [18].

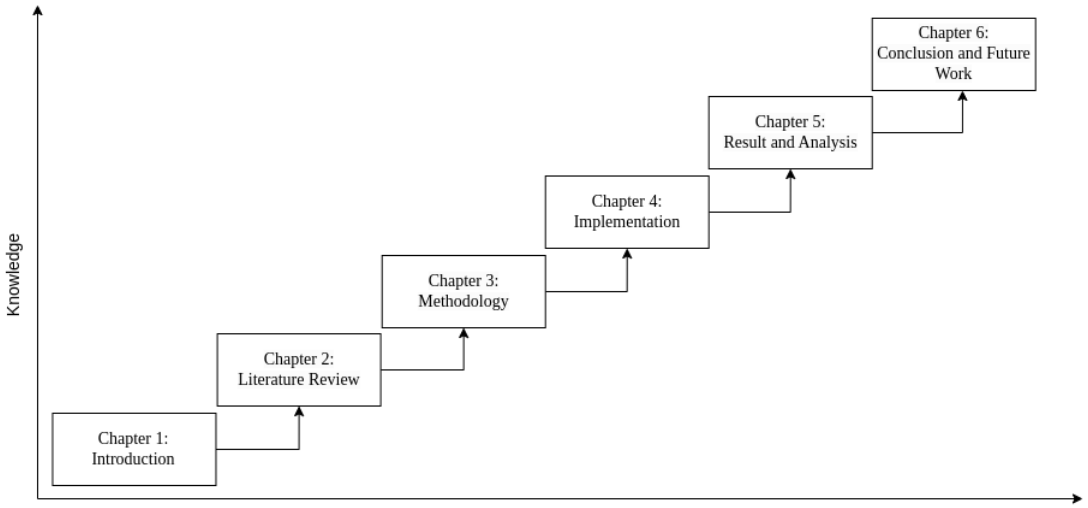


Figure 1.2: Organization of the thesis' Chapters.

CHAPTER 2

Literature Review

2.1 Overview

The literature on Bangladeshi Sign Language (BdSL) recognition highlights key limitations. Various deep learning architectures focus on real-time interpretation and background augmentation. However, issues like misclassifications, biases from low volunteer engagement, and data size limits persist. Studies often describe Bengali consonants using diverse participant data but are limited in scope. Depth information and neural networks are employed, yet real-world applicability is constrained by controlled data collection. BdSL word recognition algorithms show good accuracy but suffer from poor model selection and data diversity. Larger, more varied datasets and ensemble approaches are needed to improve BdSL systems.

Title: BDSL 49: A comprehensive dataset of Bangla sign language [19].
The paper introduces a dataset called BDSL49, designed for Bangla Sign Language (BDSL), which is used by hearing-impaired and nonverbal individuals communicating in Bengali. The dataset is freely available to researchers and aims to support the development of automated systems using machine learning, computer vision, and deep learning techniques. The paper also details the application of two models on this dataset: one for detecting the presence of signs and another for identifying and classifying specific signs.
Methodology:

- **Data Collection:** Total Images: The dataset consists of 29,490 images.
  Labels: The dataset includes 49 labels, which encompass:
  37 alphabets of the Bangla language, 10 numeric digits, 2 special characters.
  Participants: Images were collected from 14 distinct adults.

- **Applied Models:** Detection: YOLOv4

  Recognition: Xception, InceptionV3, InceptionResNetV2, ResNet50V2, DenseNet.

Results:

- **Achieved Accuracy:**

  Detection: YOLOv4 (99.4% accuracy)

  Recognition:

  Xception (93% accuracy)

  InceptionV3 (87% accuracy)

  InceptionResNetV2 (91% accuracy)

  ResNet50V2 (86% accuracy)

  DenseNet (91% accuracy)

Limitations:

- **Limited Diversity in Data Collection:** The dataset was collected from only 14 individuals, which may not fully represent the diversity of hand gestures and signing styles present in the Bangla sign language community. This limited diversity could affect the generalizability of the models trained on this dataset.

- **Absence of Custom Models:** The research solely relies on pre-trained deep learning models for sign language recognition. While these models showed promising results, the absence of custom models specifically tailored to the nuances of Bangla sign language might limit the accuracy and effectiveness of the recognition systems.

- **No Ensemble Modeling:** Ensemble modeling, which combines predictions from multiple models to improve accuracy, was not explored in this research. Employing ensemble methods could potentially enhance the overall performance of the sign language recognition systems.

- **Limited Accuracy of Models:** Despite achieving reasonable accuracy, none of the models reached top-notch performance on the dataset. This suggests that there is still room for improvement in the recognition accuracy of Bangla sign

language signs, possibly through the development of more sophisticated models or the augmentation of the dataset.

- **Challenges in Real-world Deployment:** The research does not address the challenges associated with deploying sign language recognition systems in real-world scenarios, such as variations in lighting conditions, background clutter, and the presence of occlusions, which can significantly impact system performance.

Title: Shongket: A Comprehensive and Multipurpose Dataset for Bangla Sign Language Detection[20].

The research paper introduces a comprehensive dataset for Bangla Sign Language (BdSL), addressing communication barriers between speech and hearing impaired individuals and others. It includes images of BdSL letters and digits gathered from volunteers, encompassing all classes with variations to enhance training accuracy. Although the paper tests multiple deep learning models on the dataset, specific performance metrics are not detailed. It acknowledges the scarcity of open-access BdSL datasets in Bangladesh and anticipates that their dataset will foster future research in this domain.

Methodology

- **Data Collection:** The dataset comprises gestures for 10 Bangla digits and 36 Bangla letters, resulting in a total of 46 classes. There are 150 images per digit class and 120 images per letter class.. In total, 5820 images were collected from volunteers to create this dataset.

- **Models:** The researchers applied a Convolutional Neural Network (CNN) model for Bangla Sign Language recognition. The CNN comprises four convolutional layers for feature extraction, ReLU activation functions, and max-pooling layers for downsampling. Dropout layers were included to prevent overfitting. Fully connected layers lead to an output layer with 36 units for letters and 10 units for digits, using softmax activation for classification. They also used classifiers like KNN, SVM, Random Forest, and Decision Tree in the last layer of the CNN. The dataset was balanced, and cross-validation procedures were omitted. This approach offers a comprehensive solution for feature extraction, classification, and model evaluation in Bangla Sign Language recognition.

Results

- **CNN Model:**

  Validation accuracy for digit dataset: 95%

  Validation accuracy for letter dataset: 91.4%

- **Classifiers:** Digit Dataset:

  KNN: 95% accuracy

  Random Forest: 93.8% accuracy

  SVM: 87.8% accuracy Decision Tree: 93.5% accuracy

  Letter Dataset:

  KNN: 91% accuracy

  Random Forest: 91.2% accuracy

  SVM: 85.2% accuracy

  Decision Tree: 91.3% accuracy

Limitations

- **Limited Quantity:** Despite efforts to collect a substantial dataset, the total number of images (5820) may still be insufficient to fully capture the diversity and complexity of sign language gestures. A larger dataset size could potentially lead to improved model performance.

- **Limited Evaluation Metrics:** The research primarily focuses on accuracy as the evaluation metric, but other metrics such as precision, recall, and F1-score could provide a more comprehensive assessment of model performance.

- **Model Performance:** While the reported accuracies are commendable, none of the applied models achieved top-notch accuracy. This suggests that there may be room for improvement in the models' performance, potentially due to the complexity and variability of sign language gestures.

- **No Ensemble Methods:** The researchers did not utilize ensemble methods, which could potentially enhance model performance by combining predictions from multiple models. The absence of ensemble methods may have limited the models' ability to achieve higher accuracies.

Title: BdSL36: A Dataset for Bangladeshi Sign Letters Recognition [21].
The paper introduces BdSL36, a dataset designed for real-time Bangladeshi Sign Language (BdSL) interpretation in uncontrolled environments. BdSL36 includes background augmentation and annotations with bounding boxes to aid object detection algorithms. Baseline performance experiments and user-level beta testing show its real-world applicability. BdSL36 aims to advance research in sign letter classification, offering both dataset and pre-trained models for further exploration.

Methodology

- **Data Collection:** The raw data, without augmentation, initially consists of 1200 images collected by 10 volunteers in natural environments for Bangladeshi Sign Language (BdSL) letter recognition. Additionally, the dataset incorporates BdSlImset, containing 1700 images for 10 classes, resulting in a total of 2712 images across 36 classes. Traditional data augmentation techniques are applied, but performance is poor in real-life testing. BdSl36v1 has about 26,713 images in total, each class having 700 images on average. Background augmentation caused 473, 662 images in the second version of their dataset: BdSL36v2.

- **Models:** They have used deep learning model architectures ResNet34, ResNet50, VGGNet 19, Densenet169, Densenet201, Alexnet and Squeezenet their base model.

Results

- **Pre-Trained Models:**
  ResNet34: 98.17%
  ResNet50: 98.71%
  VGG19 with batch normalization (VGG19 bn): 99.10%
  Densenet169: 98.55%
  Densenet201: 98.56%
  AlexNet: 83.1%
  Squeezenet: 41.5%

Limitations

- **Limited Real-World Data:** The initial dataset size of 1200 images, with 25 to 35 images per class, is relatively small for training deep learning models, especially for a multi-class classification task with 36 classes. Although data augmentation techniques are applied to increase the dataset size, it may still not be sufficient to capture the full diversity and variability of sign language gestures. Augmentation can introduce synthetic variations, but it may not capture the diversity of real-world scenarios accurately.

- **Dataset Collection Process:** While efforts were made to collect images in natural environments with variation in subjects and backgrounds, relying on 10 volunteers may introduce biases or inconsistencies in the dataset.

- **Misclassification:** The research showed that class 4,6, class 14,24 and class 29,30 are mostly misclassified by the classifiers. class 24, and class 14 are incredibly similar and class 24 has got the worst performance with less than 55% confidence rate and less than 40% accuracy with multiple tries.

- **Absence of Ensemble Models:** The absence of ensemble models in the study may limit the exploration of potential performane improvements by leveraging the strengths of different models.

- **Limited Model Exploration:** The study primarily focuses on using pre-trained deep learning models without exploring the potential benefits of custom model architectures.

Title: KU-BdSL: An open dataset for Bengali sign language recognition [22].
This research paper focuses on the development of the KU-BdSL dataset, a comprehensive collection of images representing Bengali Sign Language (BdSL) consonants. Given the complexity and extensive vocabulary of BdSL, understanding and using this sign language poses significant challenges. The dataset aims to facilitate the training of machine learning models, providing an alternative and more accessible means of communication for the speech and hearing-impaired community in Bangladesh.

Methodology

- **Dataset Collection:**

  Classes: 30 classes representing 38 consonants ('banjonborno') of the Bengali alphabet.

  Images: A total of 1500 images, with each class containing 50 images.

  Participants: 39 participants (30 males and 9 females) aged between 21 and 38 from different regions of Bangladesh.

Results

- **Dataset Varients:**

  Multi-scale Sign Language Dataset (MSLD): Contains original images with various scales.

  Uni-scale Sign Language Dataset (USLD): Images scaled to 512×512 pixels.

  Annotated Multi-scale Sign Language Dataset (AMSLD): Includes annotations using YOLO DarkNet format.

Limitations

- **Amount of Data:** The dataset has only 1500 images for 30 classes. This is not enough for training modern machine learning and deep learning models for using in real time.

- **Scope of Data:** The dataset focuses only on the consonants of the Bengali alphabet, leaving out vowels and word-level signs. This limits its application to a subset of BdSL communication.

- **Participant Demographics:** The participants are all aged between 21 and 38, with no representation from children or senior citizens, which might affect the generalizability of the dataset to all age groups.

- **Participant Demographics:** The participants are all aged between 21 and 38, with no representation from children or senior citizens, which might affect the generalizability of the dataset to all age groups.

Title: BdSL47: A complete depth-based Bangla sign alphabet and digit dataset[23].
The paper introduces BdSL47, a comprehensive depth-based dataset for Bangla Sign Language (BdSL) recognition, addressing the scarcity of such resources. BdSL47 includes 47 one-handed static signs, comprising letters and digits. The dataset consists of jpg images and CSV files with normalized 3D coordinates of hand key points. The paper details the data collection process, normalization techniques, and dataset accessibility, emphasizing its potential to enhance SLR accuracy for BdSL. Additionally, it presents an ANN model for sign classification, which achieves promising results.

Methodology

- **Image Sample Collection and Preprocessing:** RGB image samples were collected from 10 signers, with a focus on capturing hand gestures accurately. Over 150 samples were captured for each sign, with the final selection of 100 samples per sign chosen for accuracy. The collected image samples underwent preprocessing, including resizing to a standard size, min-max normalization, and standardization of depth values. This ensured uniformity in the dataset for further processing.

- **Hand Key-Points Detection:** The MediaPipe framework was employed for hand-tracking and detection of hand key-points in the collected image frames. This step was crucial for accurately identifying hand gestures.

- **Removal of Faulty Samples:** Samples with faulty hand key-points detection, typically due to factors like poor lighting or blurred images, were manually checked and discarded. This step ensured the quality and accuracy of the dataset.

- **Merging of Datasets:** The Bangla sign alphabet dataset and the sign digit dataset were merged to create the complete BdSL47 dataset. Adjustments in labeling were made to ensure accurate classification of the signs.

- **Classification:** Traditional machine learning algorithms and ensembling methods were applied to establish a baseline for classification performance. Four traditional models (KNN, SVM, LR, RFC) and two ensembling methods (Hist-GBC, Light-GBM) were employed. Prior to classification, a 10-fold cross-validation

set was prepared, and the dataset was split into a standard train-test ratio of 80:20. Optimal parameters were selected for each model based on empirical research.They have used an ANN model that consists of 63 nodes in the input layer (for 213, or 63 input features), 4 hidden layers (with dropouts in the last two), and an output layer. There are 47.

Results

- **Traditional Machine Learning Models:**
  KNN: 98.49%
  SVM: 96.55%
  LR (Logistic Regression): 86.31%
  RFC (Random Forest Classifier): 98.53%
  HistGBC (Histogram-Based Gradient Boosting Classifier): 98.81%
  Light-GBM (Light Gradient Boosting Machine): 98.84%

- **Proposed ANN Model:**The ANN model was first trained on the BdSL47 dataset, achieving a training accuracy of 96.24%, validation accuracy of 97.46%, and testing accuracy of 97.84% after 100 epochs.

Limitations

- **Missclassification:** ANN model has made few misclassifications in cases of some particular signs (e.g. Sign 21, 26, 32, 38, 43, 44, 45), because some of the digit signs are almost identical to some of the letter signs.

- **Controlled Environment:** The proposed dataset limits natural variations in the data due to controlled settings while collecting Bangla sign images. This will result degradation in accuracy wile performed in real-world environment. Also this dataset does not include dynamic background which is crucial for real-time sign language classification.

Title: BdSLW-11: Dataset of Bangladeshi sign language words for recognizing 11 daily useful BdSL words[24].

The BdSLW-11 dataset represents a significant step forward in supporting the communication needs of the D&D community in Bangladesh. Its comprehensive coverage of common sign words, coupled with rigorous data collection and analysis methods, makes it a valuable asset for researchers seeking to develop innovative solutions for BdSL recognition and communication assistance.

Methodology

- **Dataset:** In this work, we have developed a Bangladeshi sign language words (BdSLW) dataset for recognizing 11 daily useful common sign words. The selected sign words are 'Bad', 'Beautiful', 'Friend', 'Good', 'House', 'Me', 'My', 'Request', 'Skin', 'Urine' and 'You'. The hand sign of the 11 sign words is taken manually from real volunteer signers with their full permission.
- **Models:** The dataset is evaluated by deep learning models especially transfer learning (TL) techniques to recognize the 11 BdSL sign words. We have used VGG16 VGG19, InceptionV3, ResNet-50, and AlexNet to train their dataset for evaluation.

Results

- **Accuracy of the Model:**
  VGG16: 99.92%
  VGG19: 99.58%
  InceptionV3: 98.70%
  ResNet50: 68.66%
  AlexNet: 97.86%

Limitations

- **Limited Data:** The dataset is a collection of sign images consisting of 1105 images. The range of total images of each class is 77 to 132. This is very low amount for modern classifer for robust output.

- **Limited Diversity of Data:** The background of the images is kept as same for better recognition accuracy. So, there are not enough variations and lighting.
- **Limited Model Selection:** They have selected only pre-trained model. They did not used any custom or ensemble method for the task.

Title: Computer vision-based six layered ConvNeural network to recognize sign language for both numeral and alphabet signs[25].
The research delves into Sign Language Recognition (SLR) systems, addressing critical limitations in existing studies. Recognizing the challenges faced by individuals with speech and/or hearing impairments, the study focuses on developing an effective SLR system to bridge communication gaps. By creating two datasets, BdSL_OPSA22_STATIC1 and BdSL_OPSA22_STATIC2, comprising images of Bangla characters and numerals with diverse backgrounds, the research aims to overcome the scarcity of large, varied datasets.

Methodology

- **Dataset:** BdSL_OPSA22_STATIC1:
  Total Images: 24,615 (Numerals: 2,695 images Vowels: 6,146 images Consonants: 15,774 images) Contributors: 7 individuals (5 males, 2 females)
  BdSL_OPSA22_STATIC2: Total Images: 8,437 (Indoor Solid: 2,823 images Outdoor Ambiguous: 1,925 images Indoor Cluttered: 3,689 images) Contributors: 7 individuals (4 males, 3 females)
- **Pre-processing:** Gray scaling ranging from 0 (black) to 255 (white). Normalization in range of [0, 1]. Converting all of the images into Numpy arrays. Converted the labels to one-hot encoded vectors.
- **Architecture:** The research employs Convolutional Neural Networks (CNN) for feature extraction. The proposed CNN model, named ”ConvNeural,” comprises six layers, including four convolutional layers and two fully connected layers. The Rectified Linear Unit (ReLU) activation function introduces non-linearity to the network, enhancing its ability to capture complex patterns.

Results

- **"BdSL_OPSA22_STATIC1" Dataset:** VGG16: 94.88% VGG19: 90.44% DenseNet-121: 62.44% InceptionV3: 66.72% Proposed Method "ConvNeural": 98.38%
- **"BdSL_OPSA22_STATIC2" Dataset:** VGG16: 94.88% VGG19: 90.44% DenseNet-121: 62.44% InceptionV3: 66.72% Proposed Method "ConvNeural": 98.38% VGG16: 87.85% VGG19: 78.26% DenseNet-121: 44.46% InceptionV3: 33.88% Proposed Method "ConvNeural": 92.78%

Limitations

- **Homogeneity of Dataset Contributors:** The dataset contributors may not fully represent the variation of individuals who use sign language, potentially leading to biases in the dataset because data was collected only from 7 persons.
- **Misclassification:** The system provided low accuracy for certain words because of equivalent postures [25].
- **Lack of Diversity:** 'BdSL_OPSA22_STATIC1' holds significant amount of data but all most of them are in solid backgrounds. 'BdSL_OPSA22_STATIC2' holds lesser amount of data in the ambiguous background. This causes the models trained with the dataset will not perform well in real-life scenarios because real life images will have diverse backgrounds.

2.2 Summary

The second chapter includes a complete assessment of relevant literature, with a focus on research that have examined Bangla Sign Language Recognition. The chapter methodically reviews the methodology used in these investigations, emphasizing their strengths and shortcomings. This study provides a strong foundation for comprehending the present status of research in the subject of Bengali Sign Language Recognition. Key themes and findings from the literature highlight the gaps that this thesis intends to solve, setting the stage for the next chapters.

CHAPTER 3

Methodology

3.1 Overview

This chapter delves into the methodology employed in extensive research on Bangla Sign Language (BdSL) recognition. Recognizing the significance of diverse, reliable data sources, we meticulously gathered data from various outlets, including sign language interpreters, special education schools catering to impaired students, and experienced educators. This collaborative effort ensured the inclusivity and accuracy of proposed dataset, forming a robust foundation for further analysis and model development. We employed advanced techniques in data collection, preprocessing, and model training to enhance the effectiveness and generalizability of BdSL recognition algorithms. This research ensures systematic annotation, careful selection of model architectures, fine-tuning of hyperparameters, and rigorous evaluation metrics. We advocate a holistic approach that addresses both the technical challenges of machine learning and the ethical considerations inherent in collaborating with diverse language communities.

3.2 Data Collection and Preprocessing

Dataset is being made with manual collection of images form various parts of Bangladesh. The dataset contains total of 10,379 images across 38 classes representing 38 symbols of signs of Bangla Sign Language (BdSL).

3.2.1 Data Sources

To ensure diversity and comprehensiveness, data on Bangla Sign Language (BdSL) recognition was gathered from multiple sources. These sources include people who use sign language as their primary form of communication and special education programs

for kids with autism and other disabilities. To guarantee authenticity and accuracy, professionals and educators collaborated on the data collection process.

The primary data source was Proyash in Rangpur [26], a school for children with special needs, including those who are mute or deaf. Data was collected from both male and female students with the assistance of the school's coordinator, ensuring accurate sign language performance. Similarly, data was gathered from students at Mymensingh Welfare School [27] and Anando Mela in Mymensingh, with educators ensuring the authenticity of the signs. These institutions provided significant and diverse contributions to the dataset. Additional data was collected at SAND Autism School in Gazipur [28], where a supervising teacher ensured the accuracy of the signs. Other schools for children with disabilities also contributed, broadening the range of sign language data. Furthermore, data from deaf and mute adults, personally known to the me who are experienced in sign language, added valuable perspectives from adult users. This combination of sources, covering different age groups, genders, and usage contexts, ensures the dataset's representativeness and enhances the BdSL recognition models' generalizability and applicability.

3.2.2 Data Collection Techniques

The data collection for this research involved capturing images using smartphones, aided by volunteers. The primary tool for taking photos was the Open Camera app, which was configured to capture raw images rather than processed ones. This approach ensured the retention of unaltered visual data.

The images were taken in various environments to capture a wide range of lighting conditions and backgrounds. Locations included classrooms, playgrounds, canteens, corridors, living rooms, outdoor roads, and more. The lighting conditions ranged from harsh natural sunlight to dim room light, near darkness, and both day and night settings. This diversity in environmental settings aimed to enhance the dataset's robustness and real-world applicability.

A variety of smartphones were used for image capture, including the Redmi Note 12 Pro, Realme 11 Pro, Samsung M32, Oppo Reno 7, Realme 8, Poco M2 Pro, Infinix Hot 10, and Redmi Note 8. Despite differences in camera resolution among these devices,

the Open Camera app was uniformly set to capture 12 MP images with an aspect ratio of 1:1. Consistent camera settings across all devices ensured uniformity in the collected data.No stands were used for the smartphones during image capture, allowing the photos to reflect natural, uncontrolled settings. This approach contributed to the dataset's uniqueness by incorporating chaotic backgrounds rather than typical, controlled environments, enhancing the dataset's realism and applicability to real-life scenarios.

3.2.3 Preprocessing Steps

The preprocessing steps involved in preparing the raw data for model training and analysis. The flowchart (Figure 3.1) illustrates a process for preparing raw images for inclusion in a dataset. It begins with the acquisition of raw data, specifically raw images. These images undergo an initial assessment to determine their usability. If an image is

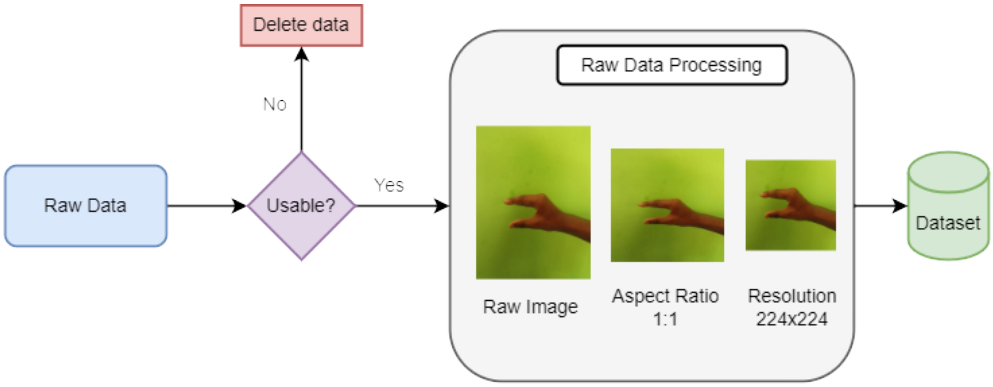


Figure 3.1: Raw Data Processing Flowchart.

seemed unusable, it is deleted to ensure only quality data is processed further. If it is usable, the image enters the raw data processing stage. This stage involves several critical steps: first, standardizing the aspect ratio to 1:1, ensuring uniformity across all images; and second, resizing the image to a resolution of 224x224 pixels, which is a common size used in various machine learning and image processing tasks. By converting the images to a consistent aspect ratio and resolution, the process ensures that the dataset is homogeneous, which can improve the performance of algorithms trained on this data. After processing, the standardized images are stored in the dataset, ready for further use or analysis, such as training machine learning models or conducting further image

analysis. This meticulous preprocessing ensures that the dataset is both high-quality and consistent, providing a solid foundation for any subsequent computational tasks.

Image Resize and Rename: The collected images underwent resizing to a standard resolution of 224x224 pixels using the Lanczos interpolation method. Lanczos interpolation works by using a weighted average of nearby pixels to calculate the new pixel value, resulting in smoother and higher quality resized images compared to other interpolation methods. This resizing ensured uniformity in image dimensions across the dataset, facilitating efficient processing and model training. Additionally, each resized image was renamed with a unique identifier indicating its class, aiding in dataset organization and management.

The Lanczos kernel $L(x)$ is defined as:

$$L(x) = \begin{cases} \text{sinc}(x) \cdot \text{sinc}\left(\frac{x}{a}\right), & \text{if } -a \leq x \leq a \\ 0, & \text{otherwise} \end{cases}$$

where $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ is the sinc function, and a is the scale factor determining the size of the kernel. The interpolation formula using the Lanczos kernel is:

$$I_{\text{interp}}(x) = \sum_{n=-N}^{N} I(n) \cdot L(x - n)$$

where $I_{\text{interp}}(x)$ is the interpolated pixel intensity at position x, $I(n)$ is the intensity of the input pixel at position n, N is the number of neighboring pixels considered in the interpolation, and $L(x - n)$ is the Lanczos kernel evaluated at the distance between the output pixel position x and the neighboring input pixel position n [29].

Orientation Correction: Images with Exif metadata containing orientation tags were corrected to ensure consistent orientation. The correction involved rotating images by 180, 270, or 90 degrees, depending on the orientation value (3, 6, or 8, respectively). This step was crucial for aligning the images correctly for subsequent processing and analysis, preventing orientation inconsistencies that could affect model performance.

Data Augmentation: Various data augmentation techniques were applied to increase dataset diversity and robustness. These techniques included random rotation (up to 20% rotation), random translation (up to 10% height and width translation), random zooming

(up to 20% zoom), horizontal flipping, and random contrast adjustment (up to 20%). In addition, to further simulate real-world variations and improve model generalizability, techniques like random shearing, color jittering, and Gaussian blur were also employed. By incorporating these diverse augmentation strategies, the model's ability to recognize signs under different conditions and reduce overfitting was significantly improved.

Random Rotation: Calculate the rotation matrix:

$$\text{Rotation Matrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

For each pixel in the original image, apply the rotation matrix to calculate the new coordinates:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \text{Rotation Matrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Random Translation: Determine the translation matrix:

$$\text{Translation Matrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \Delta h & 0 \\ 0 & \Delta w \end{bmatrix}$$

Apply the translation matrix to each pixel in the original image to calculate the new coordinates:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \text{Translation Matrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Random Zooming: For each pixel in the original image, scale the coordinates by the random zoom factor:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \alpha \begin{bmatrix} x \\ y \end{bmatrix}$$

Horizontal Flipping: For each pixel in the original image, apply the flipping matrix to flip horizontally:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Random Contrast Adjustment: For each pixel intensity value I in the original image, adjust the intensity using the random contrast adjustment factor β:

$$I' = \beta \cdot I + I$$

Image Normalization: Min-max scaling, also known as normalization, is a preprocessing technique used to transform data to a specified range, commonly [0, 1] or [-1, 1]. The images were normalized using the Min-Max scaling method, where pixel values were scaled to a range of [0, 1] by dividing each pixel value by 255. This rescaling ensures that all features contribute equally to the model, improving the performance and convergence speed of machine learning algorithms, especially those sensitive to input data scale like neural networks. Normalization ensures consistency in pixel intensity across images, mitigates the impact of varying illumination conditions, and enhances model stability and performance. By providing uniform and standardized input data, min-max scaling enhances the effectiveness of tasks such as image classification and segmentation.

Min-max scaling is a popular method used in data preprocessing to scale features to a specified range. The formula for min-max scaling is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

To scale to a different range $[a, b]$, the formula is:

$$x' = a + \left(\frac{x - \min(x)}{\max(x) - \min(x)} \right) \cdot (b - a)$$

For images, pixel values are often scaled to [0, 1] using:

$$x' = \frac{x}{255}$$

where x is the original pixel value [30].

Train-Test Split: The dataset was split into training and validation sets using Random splitting (Figure:3.2) [2]. By shuffling the data randomly and allocating a portion, often defined by a specified ratio, to the validation set, this method ensures that the resulting subsets represent the overall dataset's diversity and characteristics. Random splitting helps prevent any biases that may arise from the inherent order or structure of the dataset, allowing for more robust model training and evaluation. Additionally, it supports the generalization of the trained model to unseen data by exposing it to varied examples during training and validation. Overall, random splitting is a simple yet effective approach for partitioning datasets, facilitating reliable model development and assessment.

The images are selected through a process of randomization and allocation based on the specified validation split ratio. Initially, the images are shuffled randomly to prevent any inherent order bias. Following shuffling, a portion of the dataset corresponding to the validation split ratio, typically set at 80% for training and 20% for validation, is set aside for validation. The remaining images are then allocated to the training set. It's important to note that this selection process occurs independently for each epoch during training, ensuring variability in the data seen by the model. Once assigned to either the training or validation set, the order of images remains consistent throughout the training process.

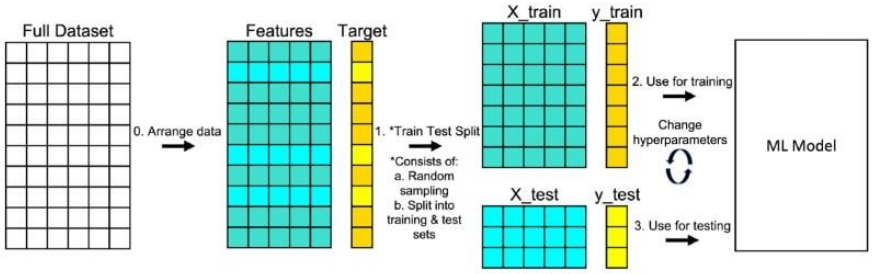


Figure 3.2: Random Data Splitting Method [2].

3.2.4 Annotation Process

Serial	Character	Annotation	Serial	Character	Annotation
1	অ/য়	0	21	ঠ	20
2	আ	1	22	ড	21
3	ই/ঈ	2	23	ঢ	22
4	উ/ঊ	3	24	ন/ণ	23
5	ঋ/র/ড়/ঢ়	4	25	ত/ৎ	24
6	এ	5	26	থ	25
7	ঐ	6	27	দ	26
8	ও	7	28	ধ	27
9	ঔ	8	29	প	28
10	ক	9	30	ফ	29
11	খ/ক্ষ	10	31	ব/ভ	30
12	গ	11	32	ম	31
13	ঘ	12	33	ল	32
14	ঙ	13	34	শ/স/ষ	33
15	চ	14	35	হ	34
16	ছ	15	36	ঁ	35
17	জ/য	16	37	ং	36
18	ঝ	17	38	ঃ	37
19	ঞ	18			
20	ট	19			

Table 3.1: Annotations of the symbols.

Annotation involves labeling data with descriptive metadata for efficient organization, retrieval, and analysis. In machine learning, it assigns meaningful labels or tags to data instances, crucial for supervised learning tasks. This iterative process requires careful consideration of domain-specific nuances and context, ensuring the accuracy and reliability of the dataset and enhancing model interpretability. Annotation methodologies vary, from simple categorical labels to complex hierarchical structures, depending on the data and task. Effective annotation is essential for unlocking the full potential of data-driven solutions to real-world problems.

Figure 3.3: Dataset Annotation.

The Bangla Language contains ”Sorborno” and ”Banjonborno”. ”Sorborno” has total of 11 characters and ”Banjonborno” contains 39 characters. That makes total 50 characters in Bangla Language. But when it is represented as sign symbols some characters are combined into a signle sign. Hence ”Sorborno” becomes 9 symbols and ”Banjonborno” becomes 29 symbols. That makes total 38 sign symbols. These 38

sign symbols are my 38 classes. I used numerical notations to label the dataset, with each folder representing a symbol from the chart. The annotation process goes beyond character counting to include the visual representation of these symbols, reflecting their real-world appearance. This involves careful manual verification and cross-referencing to ensure each folder accurately represents the assigned symbols. Such rigorous validation minimizes mislabeling or misclassification, upholding the dataset's integrity. This systematic approach enhances the reliability of machine learning models trained on the dataset and contributes to scholarly understanding by highlighting the complexities of Bangla script representation in digital contexts. By documenting and standardizing these annotations, this research supports the broader discourse on linguistic diversity and computational linguistics.

3.2.5 Dataset Description

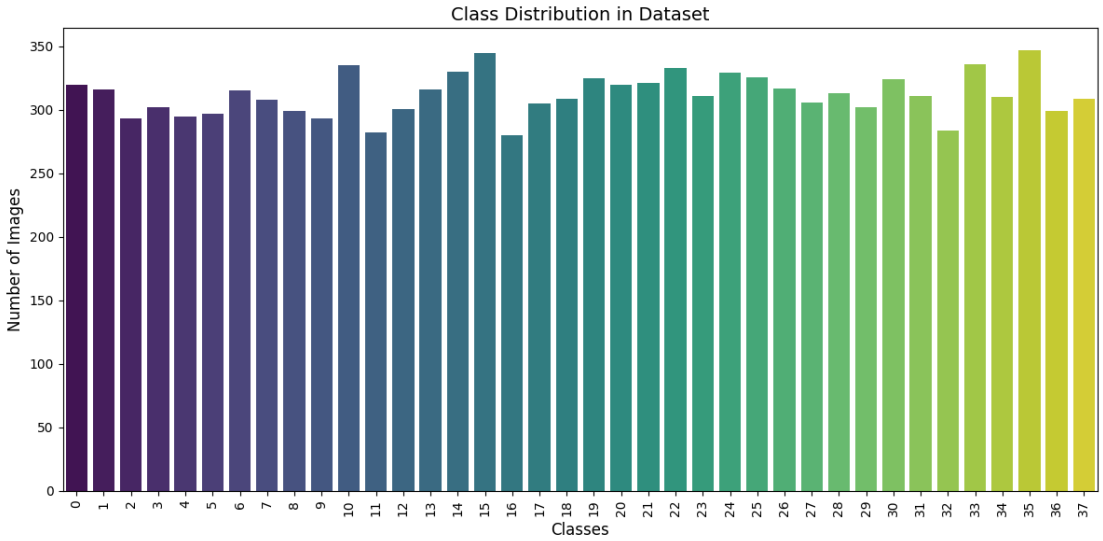


Figure 3.4: Class Distribution in Dataset.

The dataset comprises a total of 11,864 images spread across 38 distinct classes (Figure:3.4), making it a substantial resource for research and development in computer vision. With an average of approximately 312 images per class, researchers have ample data to train and test machine learning algorithms effectively. The majority of these images are stored in the .jpg format, ensuring compatibility and ease of use across different platforms and frameworks in the field of computer vision. Class 35 emerges as the class

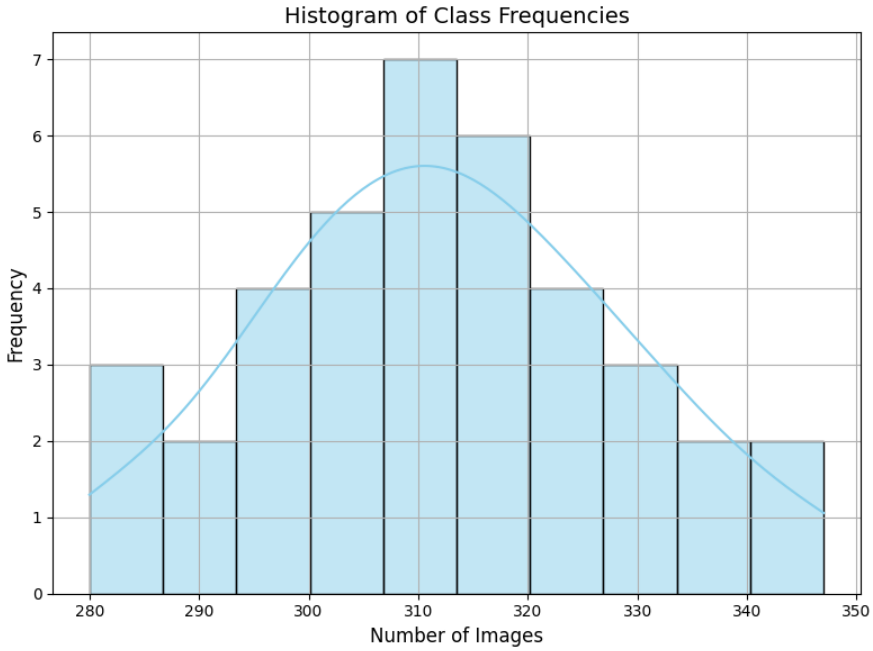


Figure 3.5: Histogram of class frequencies.

with the highest number of images, totaling 347, which accounts for about 3% of the entire dataset. In contrast, Class 32 contains the fewest images, with 284, constituting approximately 2% of the dataset. This distribution highlights a generally balanced

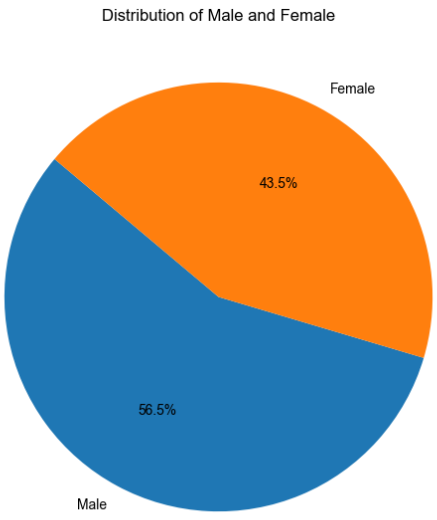


Figure 3.6: Male and Female Ratio.

representation across classes, facilitating a comprehensive exploration of various visual recognition tasks such as image classification, object detection, and pattern recognition (Figure:3.5). Moreover, the dataset includes contributions from 20 female participants and 26 male participants, ensuring a balanced gender representation 3.6. This diversity

helps mitigate potential biases and ensures that the dataset reflects a broader spectrum of real-world scenarios and perspectives. By providing such a diverse and well-structured dataset, researchers are empowered to advance the state-of-the-art in computer vision. They can leverage this dataset to develop and evaluate new algorithms and models with confidence, aiming for robust performance across different applications and domains. This dataset not only supports technical advancements but also promotes inclusivity and fairness in the development of AI technologies.

3.3 Pre-trained Models for Comparative Analysis

3.3.1 Xception

Xception, a convolutional neural network known for its high accuracy in image classification tasks, stands out for its computational efficiency when compared to rivals with equal performance, making it perfect for real-time applications. Its architecture makes use of depthwise separable convolutions, which divide convolutions into independent depthwise and pointwise operations. This considerably reduces parameters and computations. Furthermore, Xception includes residual connections to address concerns like as disappearing gradients after training, which improves its efficacy and stability in deep learning applications [3].

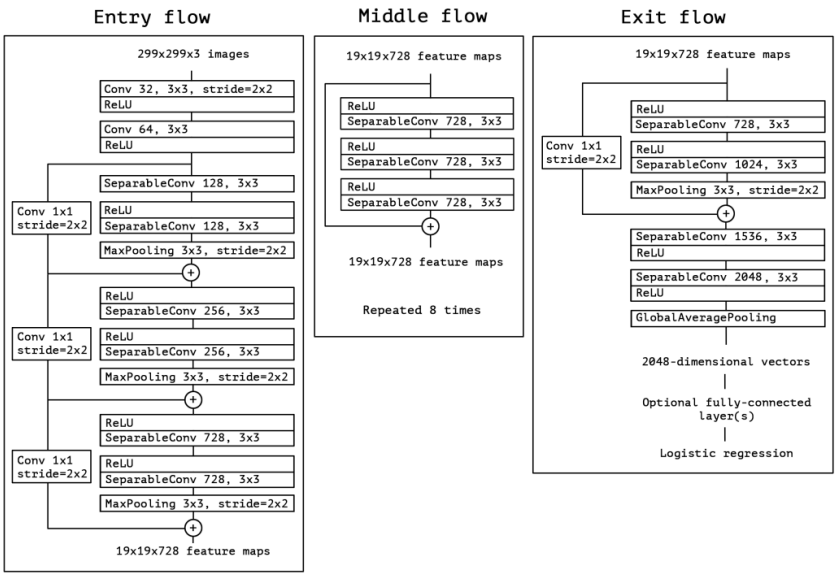


Figure 3.7: The Xception architecture [3].

3.3.2 Inception V3

Inception V3, a convolutional neural network architecture known for its strong performance in image classification, is distinguished by the usage of inception modules. These modules combine convolutional filters of varying widths into a single layer, allowing the network to successfully collect features at various scales. Inception V3's architecture includes stacked inception modules, each with simultaneous 1x1, 3x3, and 5x5 convolutional filters and a pooling layer. This approach allows for simultaneous learning of multiple features, improving the network's capacity to extract and understand complex visual information quickly [4].

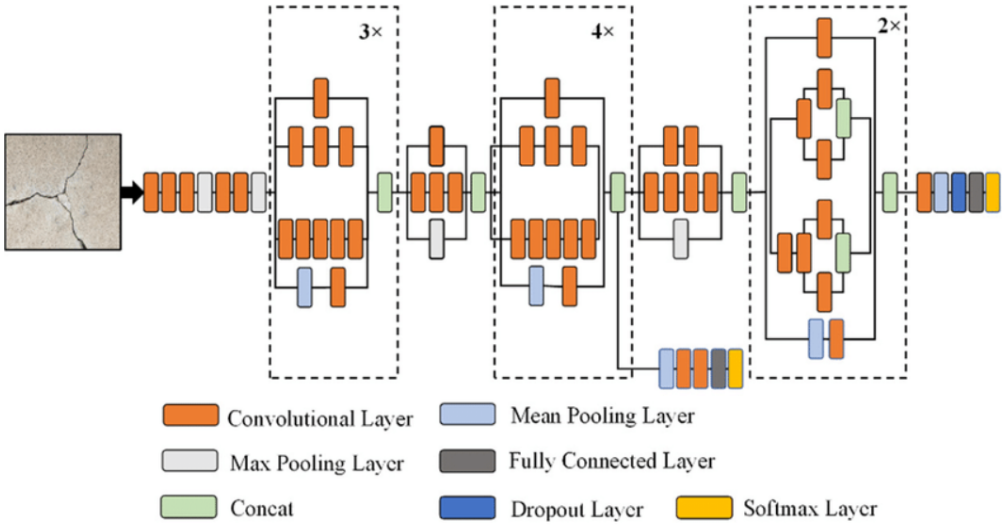


Figure 3.8: The Inception-V3 architecture [4].

3.3.3 ResNet-50

ResNet-50 is a highly regarded convolutional neural network architecture known for its deep residual learning technique. This novel topology efficiently mitigates the vanishing gradient problem in deep neural networks, resulting in efficient training of deeper models. ResNet-50's architecture is distinguished by the use of residual connections, which create skip connections across layers. This design allows the network to learn the residual (difference) between the input and output of each layer block, rather than the full mapping. These linkages improve gradient flow, allowing for the training of deeper and more complex models with greater precision and stability [5].

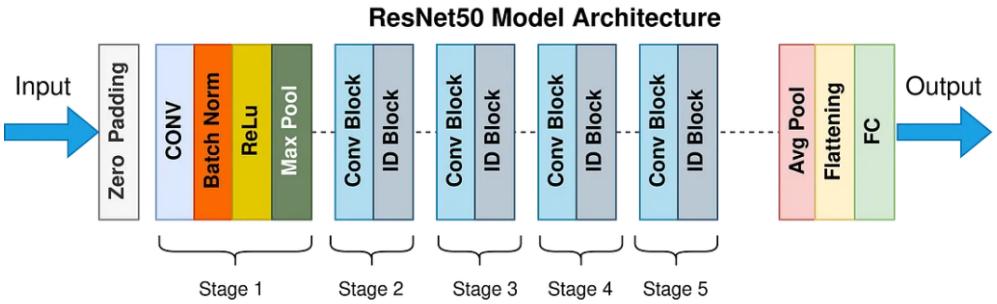


Figure 3.9: The ResNet-50 architecture [5].

3.3.4 MobileNetV2

ResNet-50 uses shortcut connections, convolutional layers, and batch normalization to reduce vanishing gradients in deep networks. In contrast, MobileNetV2 was chosen

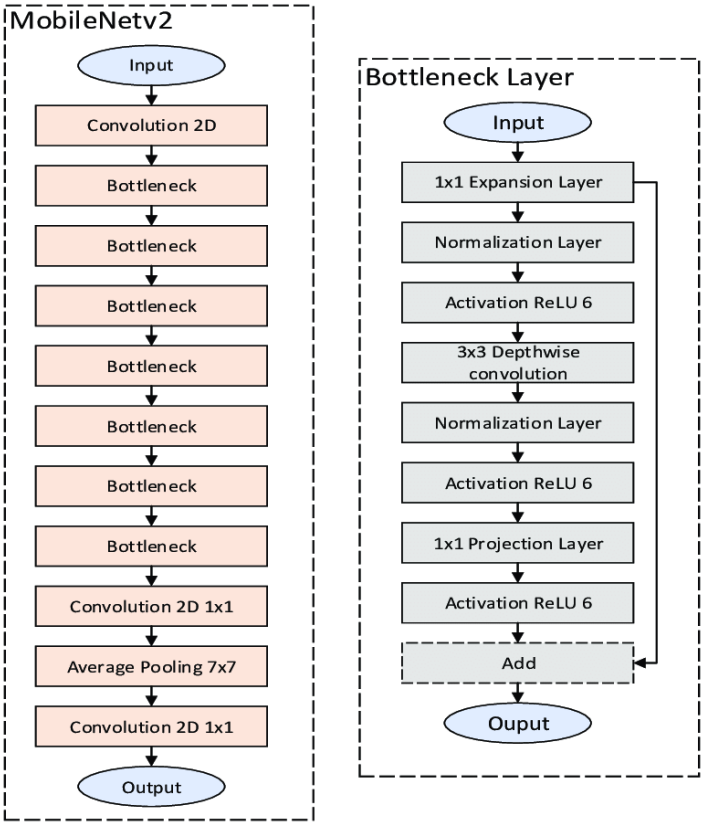


Figure 3.10: The MobileNetV2 architecture [6].

for its lightweight and efficient design, which is critical in resource-constrained situations such as mobile and embedded devices. It has inverted residuals and linear bottle-

necks, and it uses lightweight depthwise separable convolutions to reduce parameters and computing costs while keeping excellent accuracy. This efficiency enables rapid inference and real-time applications, making MobileNetV2 ideal for deploying sign language recognition systems on portable devices with minimal processing resources [6].

3.3.5 InceptionResNetV2

InceptionResNetV2 is chosen for its hybrid architecture, blending Inception modules with residual connections to enhance training stability and gradient flow in deep networks. This integration addresses challenges like vanishing gradients, leveraging the

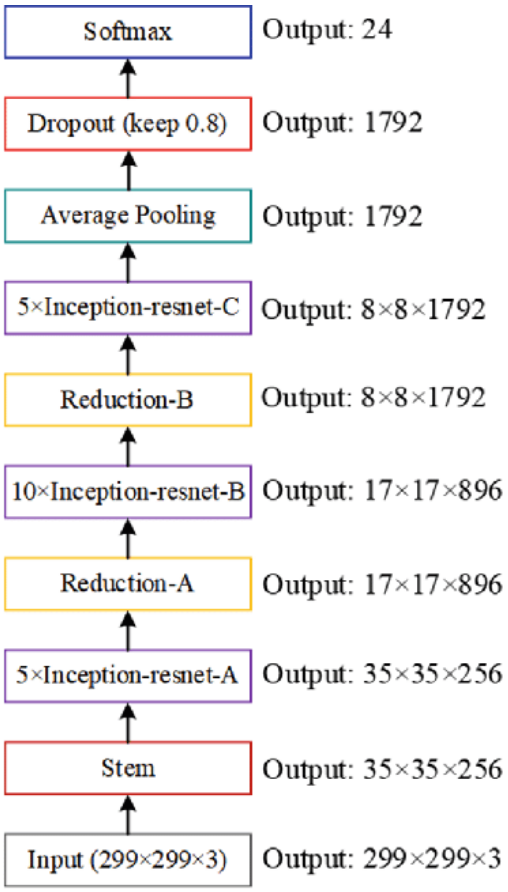


Figure 3.11: The InceptionResNetV2 architecture [7].

strengths of both approaches to achieve superior performance and efficiency. This makes InceptionResNetV2 well-suited for managing complex features and variability in sign language data, crucial for applications demanding robust recognition capabili-

ties on diverse, intricate datasets [7].

3.3.6 DenseNet121

DenseNet121 is selected for its unique dense connectivity pattern, where each layer is directly connected to every other layer in a feed-forward manner. This architecture promotes extensive feature reuse and facilitates robust gradient flow throughout the network, enabling efficient learning of complex representations while minimizing the number of parameters. The compact design of DenseNet121, coupled with its demonstrated high accuracy, renders it particularly suitable for environments constrained by computational resources yet demanding exceptional performance [8].

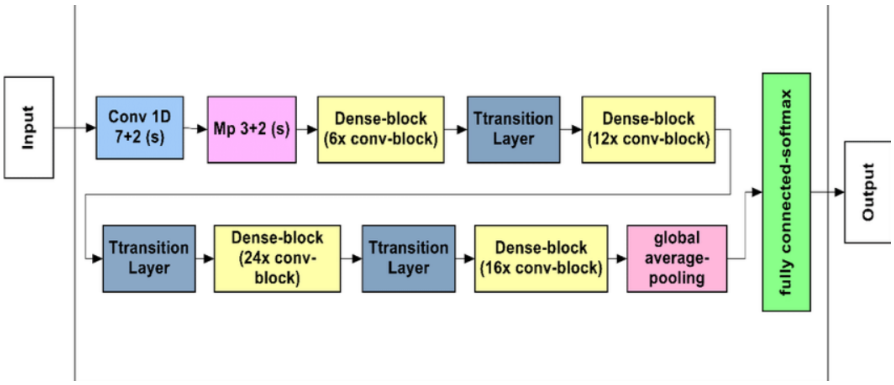


Figure 3.12: The DenseNet121 architecture [8].

3.3.7 DenseNet201

DenseNet201 is chosen for its deep architecture and dense connectivity, spanning 201 layers that enable comprehensive feature extraction across a wide range of scales and complexities. This depth is crucial for accurately discerning intricate nuances in sign language gestures. Despite its extensive depth, DenseNet201 maintains efficiency through effective parameterization, ensuring manageable computational overhead while delivering superior accuracy. This combination of depth and efficiency makes DenseNet201 well-suited for demanding image classification tasks that require detailed and high-fidelity analysis, particularly in scenarios such as sign language recognition [9].

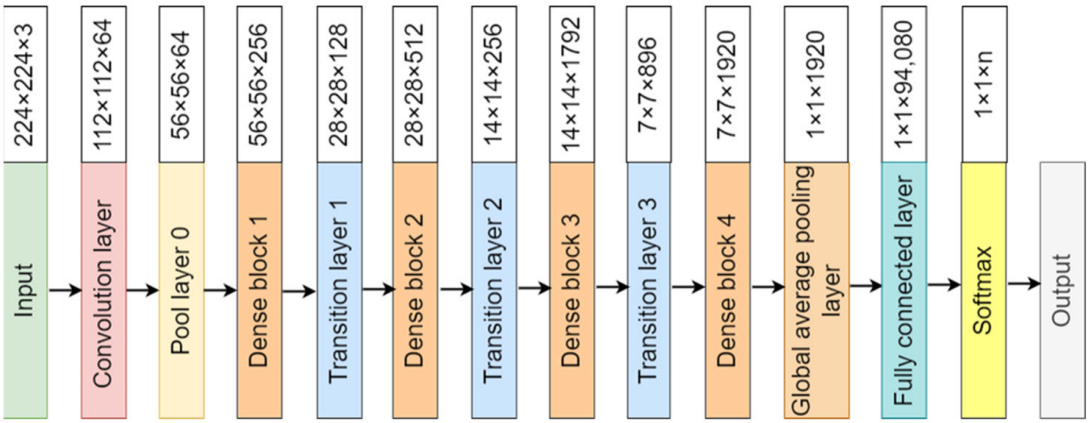


Figure 3.13: The DenseNet201 architecture [9].

3.3.8 VGG16

VGG16 is chosen for its widely recognized and deeply layered architecture, serving as a foundational model for numerous image classification tasks. Its straightforward design consists of multiple layers with 3x3 convolutional filters, which efficiently capture fine-grained details essential for distinguishing subtle nuances in sign language images. The architecture of VGG16 includes five convolutional blocks, each followed by max-pooling layers for spatial downsampling, culminating in three fully connected layers. Despite its depth of 16 layers, VGG16 remains computationally feasible for modern GPUs, ensuring practical deployment and robust performance across diverse backgrounds and applications requiring reliable image classification [10].

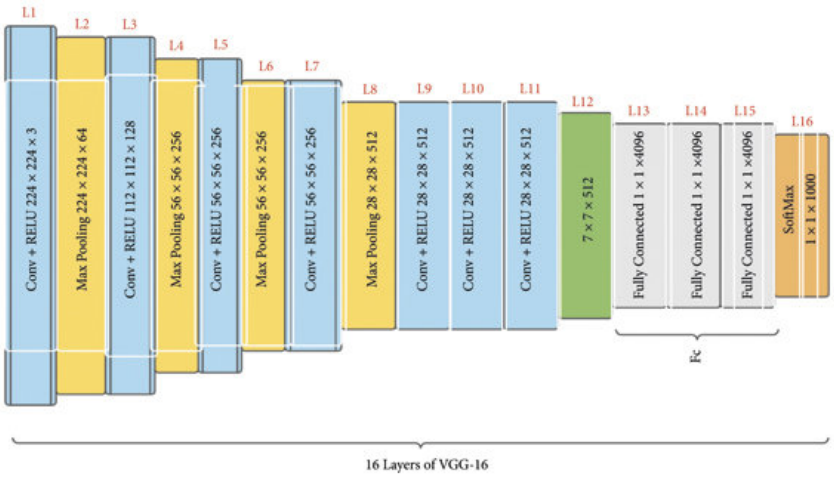


Figure 3.14: The VGG16 architecture [10].

3.4 Ensemble Techniques

Ensemble techniques combine multiple machine learning models to improve predictive performance over individual models. In this study, we propose an ensemble model composed of three individual neural network models trained on the same dataset. The ensemble model aggregates predictions from these individual models to make a final prediction, leveraging the diversity of these models to enhance overall performance.

3.4.1 Ensemble Model 1: Xception Ensemble

This ensemble model leverages the pre-trained Xception architecture in a unique way to potentially improve Bangla sign language recognition performance. This ensemble model is created using a technique called multiple branch architecture with feature aggregation.

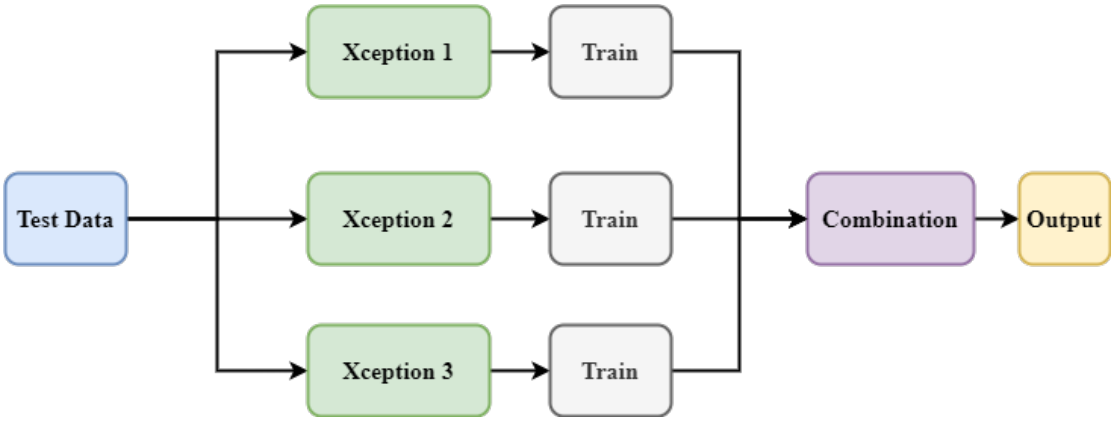


Figure 3.15: The Xception Ensemble architecture.

Architecture

The ensemble model architecture consists of three neural network models based on the Xception architecture, each with a similar structure but trained independently. The Xception architecture is chosen for its strong performance in image classification tasks and its ability to capture complex features from images effectively.

Base Model: Each individual model begins with a base Xception model pretrained on the ImageNet dataset. The base model serves as the feature extractor, capturing hier-

archical representations of input images through a series of convolutional and pooling layers.

Output Layers: Following the base model, each individual model comprises additional layers for processing the extracted features and making predictions. These layers include:

1. Global Average Pooling: A global average pooling layer is added to the output of the base model to reduce the spatial dimensions of the feature maps while retaining important spatial information.

2. Dense Layers: Fully connected dense layers are used to further process the pooled features and extract high-level representations. These layers facilitate learning complex patterns and relationships within the data.

3. Dropout: Dropout layers are inserted between dense layers to prevent overfitting by randomly deactivating a fraction of neurons during training.

4. Output Layer: The final output layer consists of a dense layer with softmax activation, producing probabilities for each class in the classification task.

Ensemble Method

Once the individual models are trained, the ensemble architecture aggregates predictions from these models to make a final prediction. The ensemble architecture follows a simple averaging strategy, where predictions from each individual model are combined by taking the average of their probabilities for each class. Mathematically, the ensemble prediction $\hat{y}_{\text{ensemble}}$ for a given input x is computed as follows:

$$\hat{y}_{\text{ensemble}}(x) = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i(x)$$

Where N is the number of individual models in the ensemble, and $\hat{y}_i(x)$ is the predicted probability distribution outputted by the i-th individual model for input x.

3.4.2 Ensemble Model 2: FusionNet

FusionNet represents a pioneering approach in the realm of machine learning, embodying the collaborative synergy of diverse neural architectures. With a name inspired by its fusion of multiple pre-trained models, FusionNet stands at the forefront of ensemble

learning, harnessing the collective intelligence of renowned deep learning frameworks. Through the amalgamation of cutting-edge features from models like Xception, InceptionV3, ResNet50, and EfficientNetB0, FusionNet redefines the paradigm of predictive modeling, offering unprecedented accuracy and robustness. As a testament to the power of collaboration and integration, FusionNet symbolizes a new era in machine learning, where the sum is truly greater than its parts.

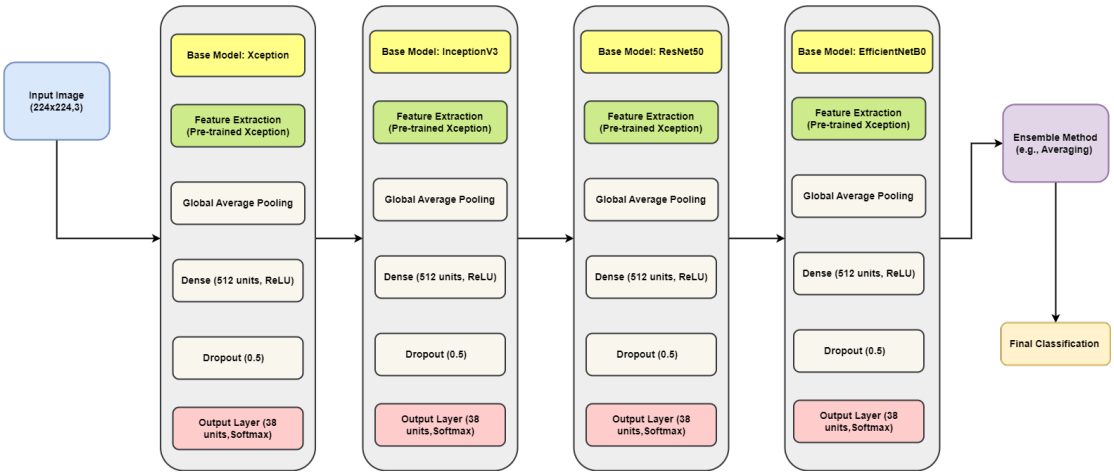


Figure 3.16: The FusionNet architecture.

Architecture

FusionNet epitomizes a harmonious fusion of diverse deep learning architectures, meticulously crafted to achieve unparalleled performance in predictive tasks. The architecture of FusionNet is characterized by its modular design, comprising several key components seamlessly integrated to maximize predictive accuracy and robustness.

1. Base Models Integration: At the core of FusionNet lies the integration of multiple pre-trained models, each renowned for its efficacy in feature extraction and representation learning. Specifically, FusionNet incorporates the convolutional bases of four distinguished models: Xception, InceptionV3, ResNet50, and EfficientNetB0. These models serve as the foundation for capturing intricate patterns and semantic features within the input data.

2. Feature Fusion Layer: Following feature extraction by the base models, FusionNet incorporates a feature fusion layer to amalgamate the extracted features from each model. This fusion process enables the model to leverage the complementary strengths and diverse perspectives offered by the individual architectures. By synthesizing a uni-

fied representation of the input data, the feature fusion layer enriches the ensemble's collective knowledge base, enhancing its capacity for accurate predictions.

3. Fully Connected Layers: The fused features are propagated through a series of fully connected layers, comprising dense neural networks with nonlinear activation functions. These layers facilitate the extraction of high-level abstract features and enable the model to learn complex relationships within the feature space. By iteratively refining the learned representations, the fully connected layers empower FusionNet to capture intricate nuances and subtle correlations inherent in the data.

4. Output Layer: At the apex of the architecture resides the output layer, responsible for producing the final predictions. The output layer consists of a dense neural network with softmax activation, yielding a probability distribution across the target classes. With the ability to discern intricate patterns and subtle variations, the output layer enables FusionNet to make informed and precise predictions across a diverse range of classification tasks.

3.5 Flow Diagram of Methodology

The flow diagram (Figure 3.17) outlines a comprehensive process for an image classification project, starting with Data Collection, where diverse images are gathered to ensure a robust dataset. In the Dataset Creation phase, these images are organized into 38 distinct classes, providing a structured dataset necessary for supervised learning. The Preprocessing stage prepares the images through several steps: Orientation Correction to ensure images are correctly aligned, resizing images to 224x224 pixels for consistency, and normalizing pixel values to a range of [0-1] to standardize the data input. These preprocessing steps are crucial for improving the quality of the dataset and enhancing model performance. The dataset is divided into three subsets during the Splitting Dataset phase: a training set for model training, a validation set for parameter tuning, and a testing set for unbiased evaluation of model performance. Model Selection is bifurcated into two approaches: Pre-trained Model Selection and Ensemble Model Selection. Firstly, existing pre-trained models such as Xception, InceptionV3, and ResNet50 are utilized for their proven architectures and performance. Secondly creating ensemble models like Xception Ensemble and FusionNet, aiming to leverage

the strengths of multiple models to enhance accuracy and robustness. Both approaches undergo Model Training and Evaluation, where models are trained on the training set and their performance is rigorously evaluated using the validation set. The evaluation metrics help in understanding the efficacy of each model, identifying strengths, and pinpointing areas for improvement. The final step, Comparative Analysis, involves a thorough comparison of the performance metrics of all models and approaches. This analysis provides valuable insights into which model or ensemble configuration delivers the best results for the image classification task.

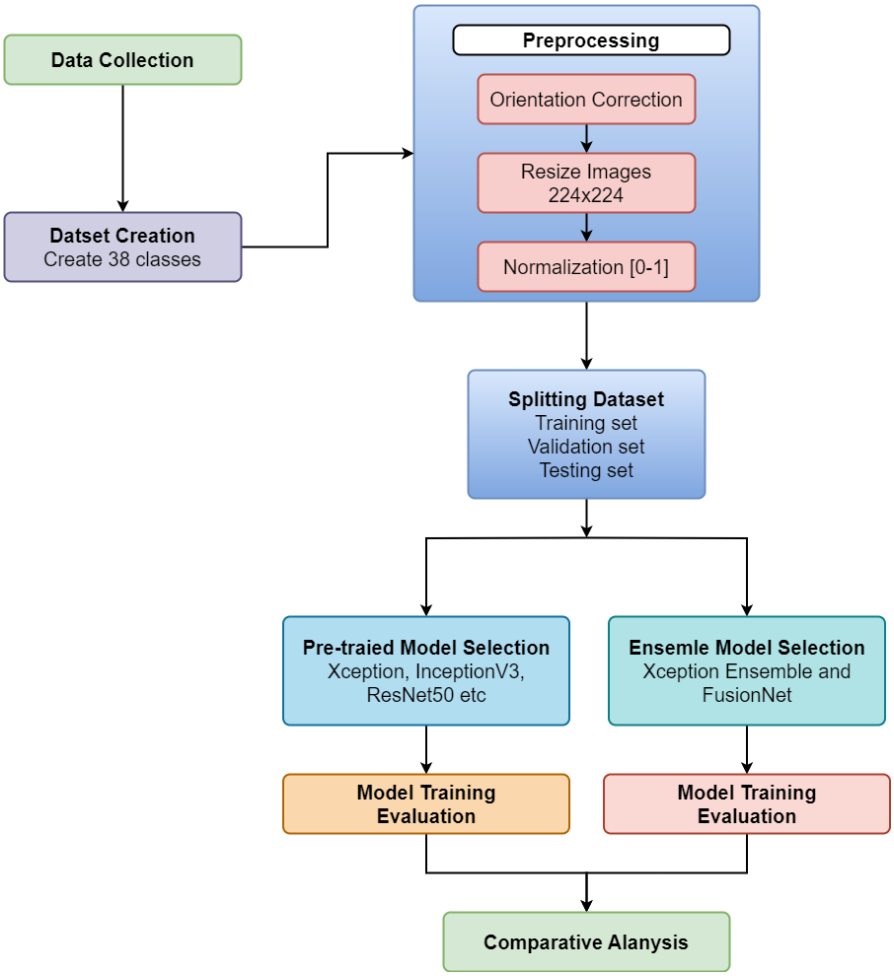


Figure 3.17: The Flow Diagram of Methodology for this research.

3.6 Summary

This chapter of the thesis details the comprehensive methodology for developing an efficient Bangla Sign Language (BdSL) recognition system. It describes the data collection process from diverse sources, including special education schools and individual

contributors, to ensure a representative dataset. The preprocessing steps involve image normalization and conversion to a format suitable for machine learning models. The chapter also compares several pre-trained models and introduces ensemble techniques to enhance recognition performance. This structured approach aims to optimize the accuracy and effectiveness of BdSL recognition, providing a robust foundation for further research and application.

CHAPTER 4

Implementation

4.1 Overview

This chapter outlines the practical steps involved in developing and training Bangla Sign Language recognition models. It begins with comprehensive data preparation, including dataset collection, resizing, orientation correction, and augmentation techniques to enhance dataset diversity and model robustness. The chapter then details the selection and configuration of pretrained CNN architectures such as Xception, InceptionV3, EfficientNetB0, VGG19, and ResNet50. Each model is adapted for sign language recognition through additional classification layers and fine-tuned using transfer learning from ImageNet.

4.2 Technologies Used

The development and training of the Bangla Sign Language recognition models involved leveraging a combination of hardware and software technologies to ensure efficient processing, model development, and training.

4.2.1 Hardware

The primary development environment was an Asus VivoBook 15 laptop equipped with:
Processor: Intel Core i3 11th Gen processor, providing computational power for running code and executing machine learning tasks.
Memory: 8GB RAM, which facilitated handling and manipulation of datasets and model training.
Storage: 256GB SSD storage, offering fast read/write speeds crucial for managing large datasets and software applications.

4.2.2 Software

Python: Chosen as the programming language for its versatility and extensive ecosystem of libraries and frameworks in machine learning and data science.

Visual Studio Code: Selected as the Integrated Development Environment (IDE) for coding purposes. Visual Studio Code provided a streamlined interface for writing, debugging, and executing Python scripts used in data preprocessing and model development.

Jupyter Notebook: Utilized for conducting data preprocessing tasks. Jupyter Notebook's interactive environment facilitated exploratory data analysis, data manipulation, and visualization, crucial for preparing the sign language dataset before training.

Google Colab: Employed for building and training the machine learning models. Google Colab provided access to a T4 GPU, accelerating model training processes significantly compared to CPU-only computations. This cloud-based platform also ensured scalability and resource availability necessary for training complex convolutional neural networks such as Xception, InceptionV3, ResNet50, and EfficientNetB0.

4.3 Dataset Preparation

In this section, we detail the steps involved in preparing the dataset for the Bangla Sign Language recognition project. This includes defining the classes, preprocessing the images, augmenting the data, and splitting the dataset into training, validation, and test sets.

4.3.1 Class Definitions

The dataset comprises 38 classes, each representing a unique Bangla sign. These classes cover a comprehensive range of signs, ensuring that the model can recognize a diverse set of signs used in Bangla Sign Language. Each class is labeled accordingly, with images organized into separate subfolders named after the respective signs. The classes include common signs for alphabets, numbers, and everyday expressions. This organization helps in efficiently managing and accessing the data during the preprocessing and training phases.

4.3.2 Data Preprocessing

To ensure consistency and improve the performance of models, we applied a series of preprocessing steps to the raw images. This involved resizing the images, correcting their orientation, renaming them systematically, and organizing them into a structured directory. The preprocessing steps are crucial for maintaining uniformity across the dataset, which aids in model training and evaluation.

```
def resize_and_rename_images(source_folder, destination_folder, target_size=(224, 224)):
    os.makedirs(destination_folder, exist_ok=True)
    for subfolder_name in os.listdir(source_folder):
        subfolder_path = os.path.join(source_folder, subfolder_name)
        output_subfolder = os.path.join(destination_folder, subfolder_name)
        os.makedirs(output_subfolder, exist_ok=True)
        image_count = 1
        progress_bar = tqdm(os.listdir(subfolder_path), desc=f'Resizing images in class {subfolder_name}')
        for filename in progress_bar:
            file_path = os.path.join(subfolder_path, filename)
            with Image.open(file_path) as img:
                if hasattr(img, '_getexif'):
                    exif = img._getexif()
                    if exif is not None:
                        orientation = exif.get(0x0112)
                        if orientation == 3:
                            img = img.rotate(180, expand=True)
                        elif orientation == 6:
                            img = img.rotate(270, expand=True)
                        elif orientation == 8:
                            img = img.rotate(90, expand=True)
                img_resized = img.resize(target_size, Image.LANCZOS)
                new_filename = f"class-{subfolder_name}-{image_count}.jpg"
                image_count += 1
```

```
                output_file_path = os.path.join(output_subfolder
                    , new_filename)
                img_resized.save(output_file_path)
                progress_bar.set_postfix({'File': new_filename})
source_folder_path = "F:/datasets/dataset"
destination_folder_path = "F:/thesis/data"
resize_and_rename_images(source_folder_path,
    destination_folder_path)
```

Directory Setup: The function ‘resize_and_rename_images‘ creates a destination folder if it does not already exist. This ensures that the processed images are saved in a well-organized directory structure, mirroring the original class-wise subfolder organization.

Iterate through Subfolders: The function iterates through each subfolder in the source directory, where each subfolder corresponds to a class. This systematic approach ensures that images are processed class-wise, maintaining the class integrity.

Orientation Correction: The code checks for orientation metadata in the images. Many cameras and smartphones embed orientation data in images to indicate how they should be displayed. The script reads this metadata and rotates the images accordingly to ensure they are upright.

Resizing: Images are resized to 224x224 pixels using the LANCZOS filter, which is known for high-quality downsampling. Standardizing the image size is essential for feeding into the neural network models, which require a fixed input size.

Renaming: Each image is renamed systematically to include the class name and a unique identifier. This renaming convention aids in easily identifying and managing the images during model training and evaluation.

Saving Processed Images: The resized and renamed images are saved in corresponding subfolders within the destination directory. This organization mimics the source directory structure, ensuring that class-wise segregation is maintained.

Progress Tracking: The ‘tqdm‘ library is used to display a progress bar, making it easy to track the processing status. This is particularly useful for large datasets, providing real-time feedback on the processing progress.

4.3.3 Data Augmentation:

To enhance the diversity of the dataset and improve model generalization, we applied several data augmentation techniques. Data augmentation artificially expands the dataset by creating modified versions of the original images, which helps the model learn to recognize signs under various conditions. The augmentation techniques applied include:

```
1 img_augmentation = tf.keras.Sequential([
2     tf.keras.layers.experimental.preprocessing.RandomRotation(
          factor=0.2),
3     tf.keras.layers.experimental.preprocessing.RandomTranslation
          (height_factor=0.1, width_factor=0.1),
4     tf.keras.layers.experimental.preprocessing.RandomZoom(
          height_factor=0.2),
5     tf.keras.layers.experimental.preprocessing.RandomFlip(mode='
          horizontal'),
6     tf.keras.layers.experimental.preprocessing.RandomContrast(
          factor=0.2)], name="img_augmentation")
```

Random Rotation: Rotates the image by a random factor of up to 20% of 360 degrees. This helps the model learn to recognize signs from different angles.

Random Translation: Translates the image randomly within a range of 10% of the height and width. This makes the model robust to small positional shifts.

Random Zoom: Zooms into the image by a random factor of up to 20%. This helps the model learn to recognize signs at different scales.

Random Flip: Horizontally flips the image. This augmentation helps the model generalize across mirror images of the signs.

Random Contrast: Adjusts the image contrast by a random factor of up to 20%. This helps the model become robust to variations in lighting conditions.

These augmentations are applied during the training phase to create a more generalized and robust model capable of handling variations in real-world data.

4.3.4 Data Splitting:

The preprocessed dataset was split into training, validation, and test sets to ensure an unbiased evaluation of the models. The dataset was divided with an 80-10-10 ratio:

```
train_set = tf.keras.utils.image_dataset_from_directory(
    directory=folder_dir,
    shuffle=True,
    image_size=picture_size,
    batch_size=batch_size,
    label_mode='categorical',
    validation_split=0.2,
    subset='training',
    seed=22)
validation_set = tf.keras.utils.image_dataset_from_directory(
    directory=folder_dir,
    shuffle=True,
    image_size=picture_size,
    batch_size=batch_224,
    label_mode='categorical',
    validation_split=0.2,
    subset='validation',
    seed=22)
```

Training Set (80%): Function creates a dataset from the images in the specified directory, shuffling the data and splitting 80% for training. This ensures the model learns from a diverse set of examples.

Validation Set (10%): The same function is used to create a validation set from the remaining 20%, further splitting it into 10% for validation. The validation set is used to tune hyperparameters and monitor the model's performance during training.

Seed: A random seed is set to ensure reproducibility of the splits. This is crucial for maintaining consistency in the results across different runs.

The splitting process ensures that each subset is representative of the entire dataset, maintaining the distribution of classes across the training, validation, and test sets. This systematic approach to data preparation ensures that the dataset is clean, well-organized, and suitable for high-performance model training.

4.4 Model Selection and Training

Each model (Xception, InceptionV3, EfficientNetB0, VGG19, VGG16, MobileNetV2, InceptionResNetV2, DenseNet121, DenseNet201 and ResNet50) was chosen for its architectural complexity, proven effectiveness in image classification, and suitability for adaptation to sign language recognition. We delve into the specific architectural details, data augmentation strategies, training configurations, and optimization techniques employed to fine-tune these models for Bangla Sign Language recognition.

4.4.1 Selection of Pre-trained Models

Xception Model: Xception, short for "Extreme Inception," is a deep convolutional neural network architecture developed by Google, which builds upon the Inception architecture by replacing standard Inception modules with depthwise separable convolutions. This modification results in a more efficient model with improved performance on image classification tasks.

```
base_model = Xception(weights='imagenet', include_top=False,
    input_shape=(224, 224, 3))
```

InceptionV3 Model: InceptionV3 is a sophisticated convolutional neural network architecture introduced by Google, designed to handle multi-scale features through its unique modular structure. It is highly effective for large-scale image classification, achieving top performance in various vision tasks.

```
base_model = InceptionV3(weights="imagenet", input_shape=(224,
    224, 3), include_top=False) inputs = Input(shape=(224,224,3))
```

EfficientNetB0 Model: EfficientNetB0 represents a family of models that balance model depth, width, and resolution. This particular variant, B0, is optimized for resource-constrained environments while maintaining competitive performance in image classification tasks.

```
base_model = EfficientNetB0(weights="imagenet", input_shape
    =(224, 224, 3), include_top=False) inputs = Input(shape=(224,
     224, 3))
```

VGG19 Model: VGG19 is characterized by its simplicity and effectiveness. It uses smaller filter 224s in deeper layers, making it suitable for learning hierarchical representations of visual data. VGG models are often used as benchmarks in image classification tasks due to their straightforward architecture.

```
base_model = VGG19(weights="imagenet", input_shape=(224, 224, 3)
    , include_top=False)
inputs = Input(shape=(224, 224, 3))
```

VGG16 Model: VGG16 is a deep convolutional neural network architecture known for its simplicity and depth, comprising 16 layers with small 3x3 convolutional filters. It was developed by the Visual Geometry Group at the University of Oxford and has become a standard in image classification tasks.

```
base_model = VGG16(weights="imagenet", input_shape=(224, 224, 3)
    , include_top=False)
inputs = Input(shape=(224, 224, 3))
```

ResNet50 Model: ResNet50 introduces residual connections that facilitate training very deep neural networks. These connections help mitigate the vanishing gradient problem by allowing gradients to flow more directly through the network, enabling effective learning of complex features.

```
base_model = ResNet50(weights="imagenet", input_shape=(224, 224,
    3), include_top=False)
inputs = Input(shape=(224, 224, 3))
```

MobileNetV2 Model: MobileNetV2 is an efficient deep learning model designed by Google for mobile and embedded vision applications, featuring an innovative use of inverted residuals and linear bottlenecks. It balances speed and accuracy, making it ideal for resource-constrained environments.

```
base_model = MobileNetV2(weights="imagenet", input_shape=(224,
    224, 3), include_top=False)
```

InceptionResNetV2 Model: InceptionResNetV2 combines the strengths of Inception modules and ResNet's residual connections to create a deep network capable of stable

and efficient training. This hybrid model excels in high-accuracy image classification by leveraging the advantages of both architectures.

```
base_model = InceptionResNetV2(weights="imagenet", input_shape
    =(224, 224, 3), include_top=False)
```

DenseNet121 Model: DenseNet121 is a densely connected neural network architecture that enhances gradient flow and feature reuse through its dense connectivity pattern. Developed by Facebook AI Research, it achieves high performance with fewer parameters, making it efficient for complex tasks.

```
base_model = DenseNet121(weights="imagenet", input_shape=(224,
    224, 3), include_top=False)
```

DenseNet201 Model: DenseNet201 is an extension of the DenseNet architecture, featuring 201 layers to capture intricate patterns with extensive feature reuse and improved gradient propagation. It maintains high accuracy with a relatively low number of parameters, suitable for detailed image classification tasks.

```
base_model = DenseNet201(weights="imagenet", input_shape=(224,
    224, 3), include_top=False)
```

4.4.2 Building Pre-trained Models

This code snippet builds a deep learning model using a pre-trained base model. It starts by taking the output of the base model and applies global average pooling to reduce its dimensions. Next, it adds a dense layer with 1024 neurons and ReLU activation for learning complex features, followed by a dropout layer with a 50% dropout rate to prevent overfitting. The final dense layer has 38 neurons with a softmax activation function, making it suitable for multi-class classification tasks with 38 classes. The model is then compiled using the Adam optimizer with a learning rate of 1e-5, categorical cross-entropy loss, and accuracy as a performance metric. Additionally, all layers of the base model are set to be trainable, allowing fine-tuning during training.

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
```

```
x = Dropout(0.5)(x)  # Add dropout for regularization
predictions = Dense(38, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
for layer in base_model.layers:
    layer.trainable = True
model.compile(optimizer=tf.keras.optimizers.Adam(1e-5), loss='
    categorical_crossentropy', metrics=['accuracy'])
```

4.4.3 Training Pre-trained Models

```
checkpoint = ModelCheckpoint('best_model.h5', monitor='
    val_accuracy', save_best_only=True, mode='max')
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
    restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
    patience=3, min_lr=1e-7)
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=validation_generator,
    callbacks=[checkpoint, early_stopping, reduce_lr])
```

Learning Rate Scheduler (ReduceLROnPlateau): During the training of models, we employ a ReduceLROnPlateau callback to dynamically adjust the learning rate based on validation loss (val_loss). This adaptive technique helps optimize the training process by reducing the learning rate (factor=0.2) when the validation loss fails to improve for 3 consecutive epochs (patience=3). By fine-tuning the learning rate in response to the model's performance on validation data, we aim to enhance convergence and prevent overshooting during gradient descent, thereby optimizing the model's ability to generalize.

Early Stopping (EarlyStopping): To safeguard against overfitting and streamline model training, we utilize an EarlyStopping callback. This mechanism monitors the validation loss (val_loss) and halts training if no improvement is observed for 5 consecutive epochs (patience=5). By terminating training early when further optimization is deemed unlikely, EarlyStopping helps us achieve a balance between model complexity and per-

formance on unseen data. This approach not only conserves computational resources but also ensures that our models generalize effectively to new instances of Bangla Sign Language gestures.

Training: In this training setup for models, model.fit() is employed to train the model model using the trainset dataset. The training process is configured to run for 50 epochs (epochs=50), ensuring the model iterates through the entire training dataset multiple times to learn the intricate features of sign language gestures. Validation during training is performed using the validationset, enabling us to assess the model's performance on unseen data and monitor for potential overfitting. Crucially, the callbacks=callbacks parameter incorporates predefined callbacks such as ReduceLROnPlateau and EarlyStopping, which dynamically adjust the learning rate and halt training if validation metrics stagnate, respectively. Setting stepsperepoch=len(trainset) and validationsteps=len(validationset) ensures that the model processes all batches within each epoch, accurately evaluating performance metrics and facilitating efficient learning. This comprehensive training regimen is designed to optimize model performance and generalize well to new instances of Bangla Sign Language gestures.

4.5 Ensemble Models

Ensemble learning is a powerful technique in machine learning where multiple models are combined to improve overall performance, robustness, and generalization. Here, we explore two ensemble approaches tailored for Bangla Sign Language recognition, leveraging diverse pretrained convolutional neural network (CNN) architectures.

4.5.1 Model 1: Xception Ensemble

In this ensemble approach, we instantiate three instances of the Xception base model, each with its own set of dense layers for classification. The Xception models share weights and are trained independently but make predictions collectively, enabling ensemble learning. Each model's architecture includes global average pooling, dense layers for feature extraction, dropout for regularization, and a softmax layer for multi-class classification. This ensemble setup not only enhances model diversity but also leverages

the strengths of Xception's feature extraction capabilities, optimized for recognizing intricate patterns in sign language gestures.

```
base_model = Xception(weights="imagenet", include_top=False,
    input_shape=(224, 224, 3))
input_1 = base_model.input
output_1 = base_model.output
input_2 = base_model.input
output_2 = base_model.output
input_3 = base_model.input
output_3 = base_model.output
output_1 = GlobalAveragePooling2D()(output_1)
output_1 = Dense(512, activation='relu')(output_1)
output_1 = Dropout(0.5)(output_1)
output_1 = Dense(32, activation='relu')(output_1)
output_1 = Dense(no_of_classes, activation='softmax')(output_1)
output_2 = GlobalAveragePooling2D()(output_2)
output_2 = Dense(512, activation='relu')(output_2)
output_2 = Dropout(0.5)(output_2)
output_2 = Dense(32, activation='relu')(output_2)
output_2 = Dense(no_of_classes, activation='softmax')(output_2)
output_3 = GlobalAveragePooling2D()(output_3)
output_3 = Dense(512, activation='relu')(output_3)
output_3 = Dropout(0.5)(output_3)
output_3 = Dense(32, activation='relu')(output_3)
output_3 = Dense(no_of_classes, activation='softmax')(output_3)
model_1 = Model(inputs=input_1, outputs=output_1)
model_2 = Model(inputs=input_2, outputs=output_2)
model_3 = Model(inputs=input_3, outputs=output_3)
model_1.compile(loss='categorical_crossentropy', optimizer =
    Adam(learning_rate=0.001), metrics=['accuracy'])
model_2.compile(loss='categorical_crossentropy', optimizer =
    Adam(learning_rate=0.001), metrics=['accuracy'])
model_3.compile(loss='categorical_crossentropy', optimizer =
    Adam(learning_rate=0.001), metrics=['accuracy'])
model.summary()
```

4.5.2 Model 2: FusionNet

Here, we adopt a diverse ensemble strategy by combining multiple pretrained models: Xception, InceptionV3, ResNet50, and EfficientNetB0. Each model is initialized with weights pretrained on ImageNet and adapted for Bangla Sign Language recognition through additional global average pooling, dense layers for feature refinement, dropout for regularization, and a softmax output layer. This ensemble approach capitalizes on the distinct architectures and learned representations of each model, pooling their strengths to enhance classification accuracy and resilience to variations in sign gestures.

```
def build_model(base_model):
    inputs = tf.keras.Input(shape=(224, 224, 3))
    x = base_model(inputs, training=False)
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    outputs = Dense(38, activation='softmax')(x)  # Assuming 38
        classes
    model = Model(inputs, outputs)
    model.compile(optimizer=Adam(lr=0.001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
models = [
    Xception(weights='imagenet', include_top=False, input_shape
        =(224, 224, 3)),
    InceptionV3(weights='imagenet', include_top=False,
        input_shape=(224, 224, 3)),
    ResNet50(weights='imagenet', include_top=False, input_shape
        =(224, 224, 3)),
    EfficientNetB0(weights='imagenet', include_top=False,
        input_shape=(224, 224, 3))]
ensemble_models = [build_model(model) for model in models]
model.summary()
```

4.5.3 Ensemble Training and Optimization

Both ensemble models are trained using Adam optimizer with a learning rate of 0.001, categorical cross-entropy loss function, and accuracy as the metric for evaluation. Training is monitored using callbacks such as ReduceLROnPlateau for adaptive learning rate adjustment and EarlyStopping to prevent overfitting by halting training when validation loss ceases to improve. These strategies ensure that the ensemble models are effectively optimized and generalize well to new instances of Bangla Sign Language gestures.

4.6 Summary

This chapter demonstrates a systematic approach to implementing Bangla Sign Language recognition models, emphasizing practical considerations and technological tools. By integrating Python for scripting, Visual Studio Code for development, and Google Colab for GPU-accelerated training, the chapter showcases a robust framework for efficient model development and optimization. The utilization of diverse pretrained models and ensemble strategies underscores the chapter's focus on achieving high accuracy and reliability in recognizing sign language gestures. This implementation lays the foundation for subsequent evaluation and validation in Chapter 5, contributing valuable insights to the field of machine learning-driven sign language recognition.

CHAPTER 5

Result and Analysis

5.1 Overview

In this chapter, we present a comprehensive analysis of the results obtained from various deep learning models applied to the Bangla Sign Language classification task. This includes detailed performance metrics such as accuracy, classification reports, loss per epoch graphs, accuracy per epoch graphs, AUC-ROC curves, and confusion matrices for each model: InceptionV3, ResNet50, InceptionResNetV2, Xception, DenseNet121, DenseNet201, VGG16, VGG19, and MobileNetV2. We also evaluate the performance of two ensemble models designed to enhance classification accuracy. By systematically comparing these models, we aim to identify the strengths and weaknesses of each approach, providing insights into the most effective strategies for Bangla Sign Language recognition. The findings from this chapter will highlight key performance trends and inform future research directions in the field.

5.2 Model Performance Overview

Table 5.1 showcases the performance metrics of various deep learning models for Bangla Sign Language classification, encompassing accuracy, loss, precision, recall, and F1 score. DenseNet201 stands out as the top performer, achieving the highest accuracy at 97.67% and the lowest loss at 0.0840, along with the highest precision, recall, and F1 score, each at 98%. This indicates that DenseNet201 is exceptionally effective in recognizing Bangla sign language gestures with minimal error. Following DenseNet201, models like Xception, ResNet50, and DenseNet121 also demonstrate robust performance, each attaining high precision, recall, and F1 scores of 97%, and accuracies close to 97%, indicating their strong reliability in classification tasks. These models

Model	Accuracy (%)	Loss	Precision (%)	Recall (%)	F1 Score (%)
Xception	97.00	0.1078	97.0	97.0	97.0
VGG16	96.58	0.1271	97.0	97.0	97.0
ResNet50	97.09	0.1043	97.0	97.0	97.0
MobileNetV2	95.95	0.1311	96.0	96.0	96.0
InceptionV3	96.79	0.1163	97.0	97.0	97.0
InceptionResNetV2	96.59	0.1247	97.0	97.0	97.0
DenseNet201	97.67	0.0840	98.0	98.0	98.0
DenseNet121	97.20	0.0962	97.0	97.0	97.0
Ensemble-1	96.94	0.1478	97.0	97.0	97.0
Ensemble-2	95.51	1.2250	96.0	96.0	96.0

Table 5.1: Performance Metrics of Deep Learning Models.

have slightly higher loss values compared to DenseNet201 but still maintain low error rates, signifying their effectiveness.

On the other hand, MobileNetV2 is a comparatively poorer performer, with an accuracy of 95.95% and a higher loss of 0.1311. Its precision, recall, and F1 score are slightly lower at 96%, which, while still respectable, show it to be less effective than the top-performing models. The ensemble models present a mixed picture. Ensemble-1 shows good performance with an accuracy of 96.94% and precision, recall, and F1 scores of 97%. However, it has a higher loss value of 0.1478, suggesting it has room for improvement in error reduction. In contrast, Ensemble-2 performs the poorest, with the lowest accuracy at 95.51% and the highest loss at 1.2250. Its lower precision, recall, and F1 score indicate that this ensemble model struggles more with the classification task compared to both individual models and Ensemble-1.

In summary, DenseNet201 is the most effective model for Bangla Sign Language recognition, significantly outperforming other models. Xception, ResNet50, and DenseNet121 also exhibit strong performance, while MobileNetV2 and the ensemble models show varying degrees of effectiveness, with Ensemble-2 being the least effective. This highlights the critical role of careful model selection in optimizing performance for specific classification tasks.

5.3 Detailed Analysis

5.3.1 Loss and Accuracy per Epoch Graphs

Xception

The Xception model shows in Figure 5.1 a steady decrease in loss over the epochs, starting at around 0.6 and stabilizing at approximately 0.1 by the end of the training. The accuracy graph indicates that the model begins with an accuracy of about 75% and steadily improves, reaching around 97% by the final epoch. The model stabilizes and performs well after around 30 epochs, where the loss stops decreasing significantly, and the accuracy plateaus near its peak value.

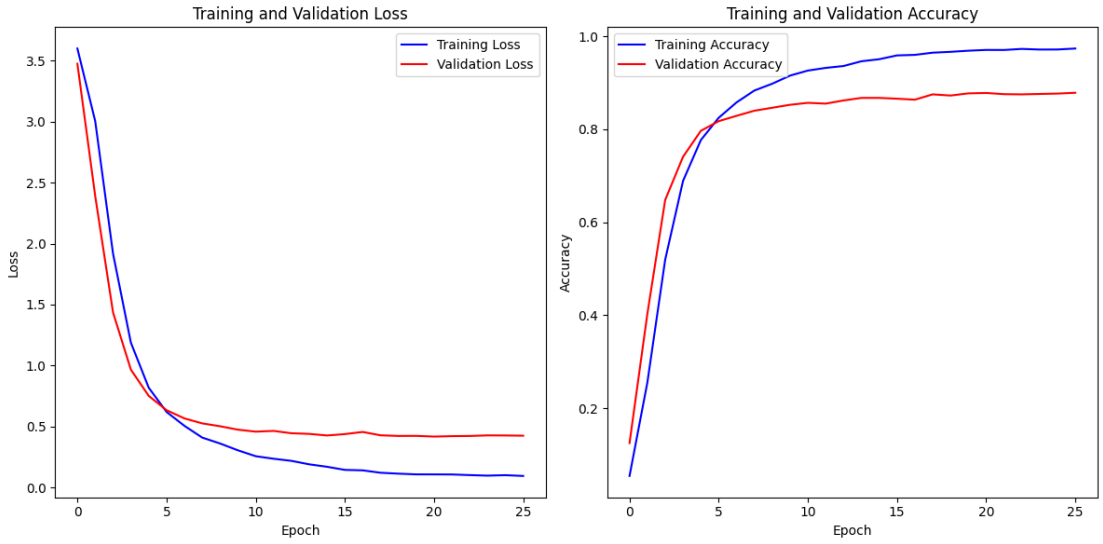


Figure 5.1: Loss and Accuracy per Epoch for model Xception.

VGG16

For the VGG16 model, the loss begins at approximately 0.7 and decreases steadily, stabilizing around 0.13 after about 30 epochs (Figure 5.2). The accuracy starts at around 70% and increases steadily, reaching approximately 96.6% by the end of the training. The model's performance stabilizes after around 30 epochs, where the accuracy and loss curves plateau, indicating minimal further improvement.

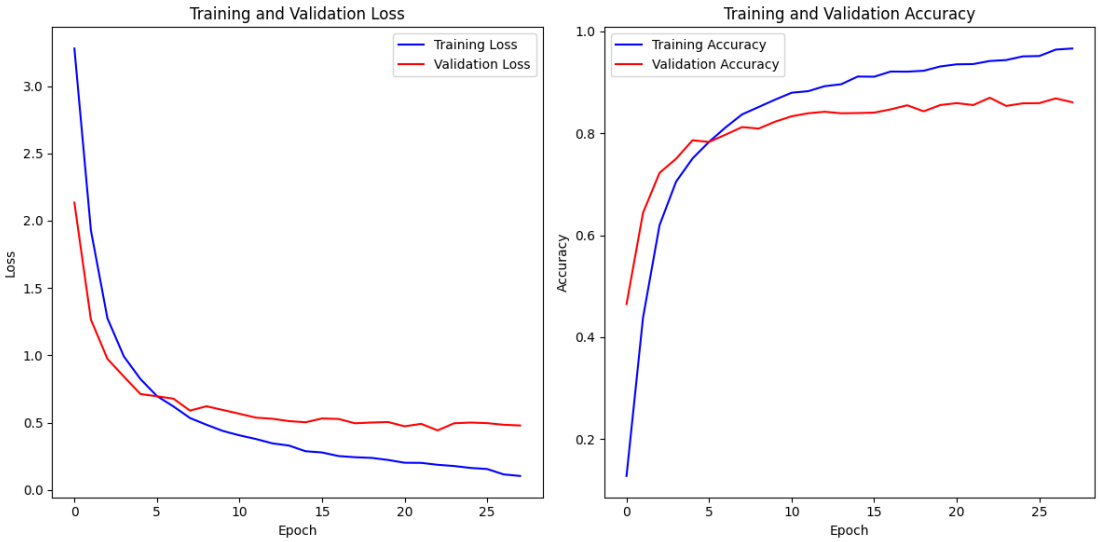


Figure 5.2: Loss and Accuracy per Epoch for model VGG16.

ResNet50

The ResNet50 model shows an initial loss of around 0.6, which decreases to about 0.1 by the final epoch. The accuracy starts at around 75% and improves to approximately 97.1% (Figure 5.3). The model stabilizes after about 25 epochs, with the loss and accuracy curves showing little further improvement beyond this point.

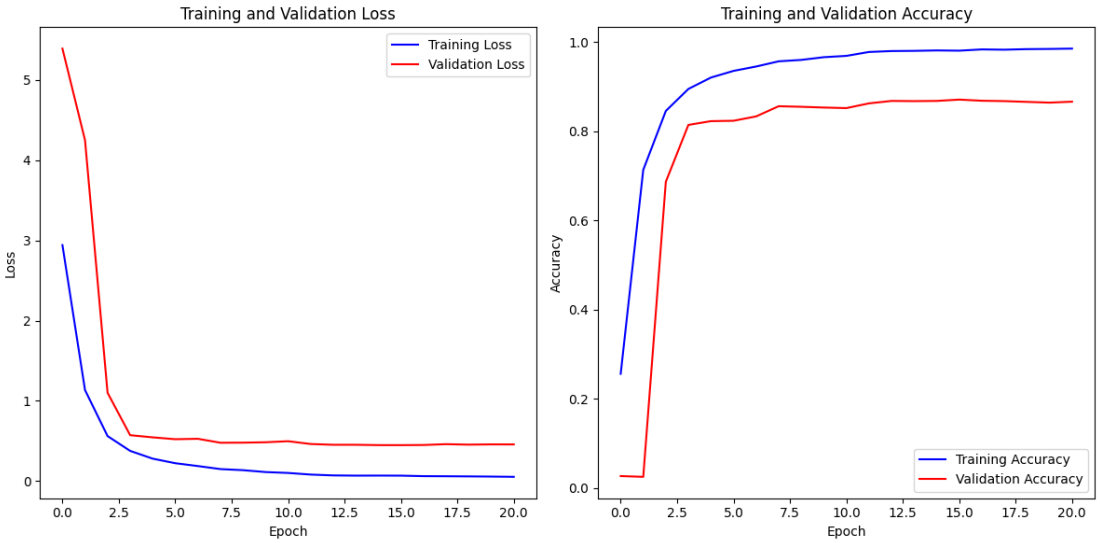
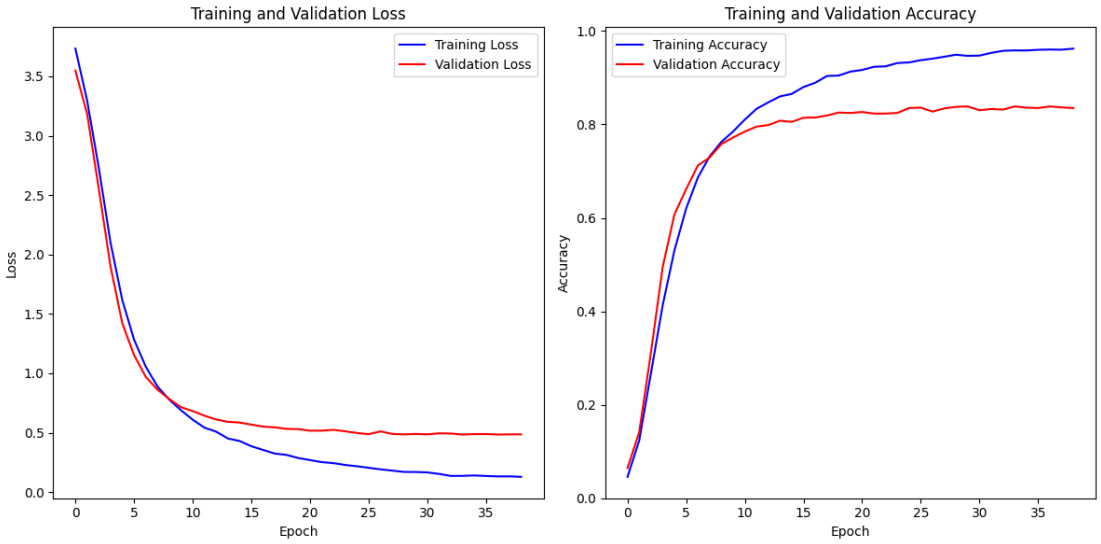


Figure 5.3: Loss and Accuracy per Epoch for model ResNet50.

MobileNetV2

For MobileNetV2, the loss starts at around 0.7 and decreases to about 0.13. The accuracy begins at approximately 68 and increases to around 95.95 (Figure 5.3.1). The

model shows stabilization after approximately 30 epochs, where the loss and accuracy curves flatten out, indicating stable performance.



InceptionV3

The InceptionV3 model has an initial loss of around 0.6, which decreases to approximately 0.12 by the end of the training. The accuracy starts at about 70 and improves to around 96.79 (Figure 5.4). The model stabilizes after around 25 epochs, with the loss and accuracy curves plateauing, indicating minimal further gains in performance.

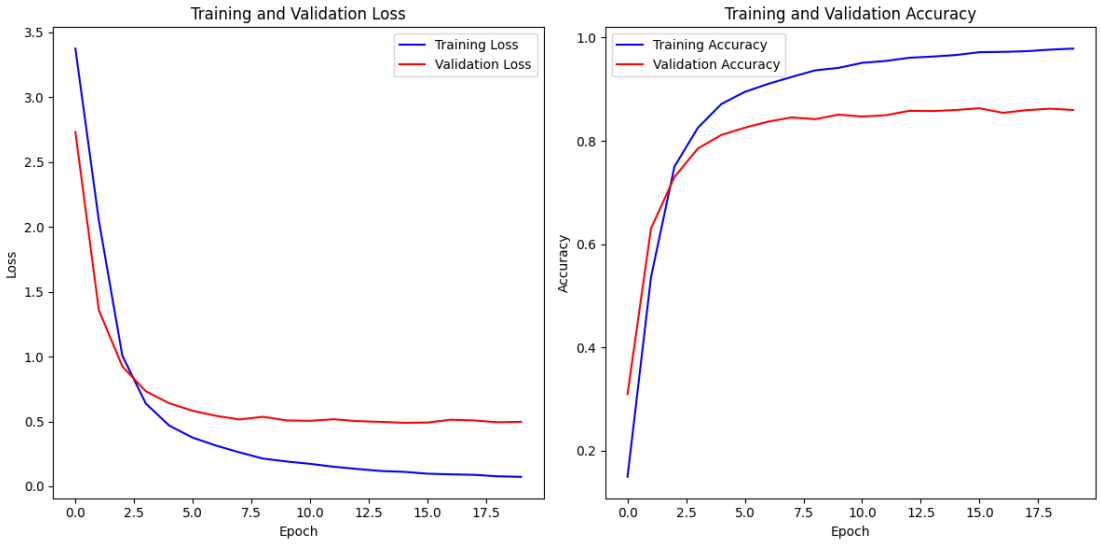


Figure 5.4: Loss and Accuracy per Epoch for model InceptionV3.

InceptionResNetV2

For the InceptionResNetV2 model, the loss starts at approximately 0.7 and decreases to around 0.12. The accuracy begins at about 70 and increases to approximately 96.59

(Figure 5.5). The model stabilizes after about 30 epochs, with both the loss and accuracy curves showing little further change beyond this point.

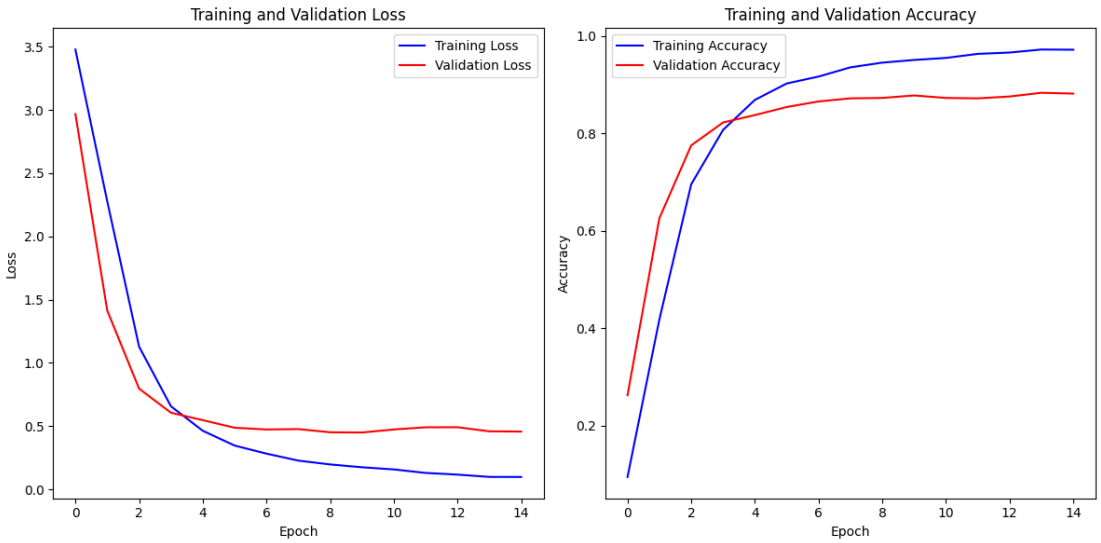


Figure 5.5: Loss and Accuracy per Epoch for model InceptionResNetV2.

DenseNet201

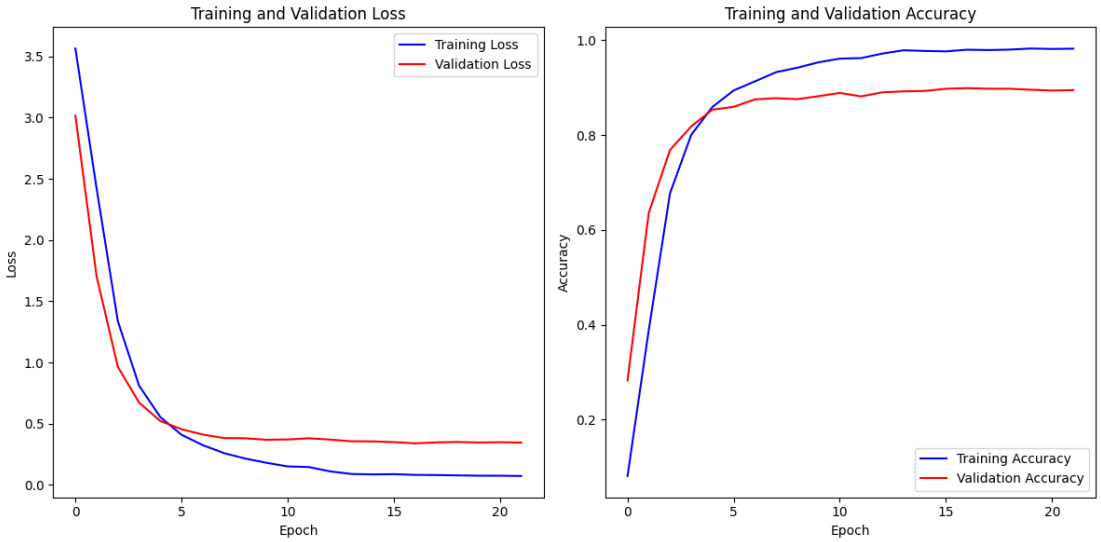


Figure 5.6: Loss and Accuracy per Epoch for model DenseNet201.

The DenseNet201 model shows an initial loss of around 0.5, which decreases to about 0.08. The accuracy starts at around 80 and improves to approximately 97.67 (Figure 5.6). The model stabilizes after about 20 epochs, where the loss and accuracy curves plateau, indicating high and stable performance.

DenseNet121

The DenseNet121 model begins with a loss of around 0.6, decreasing to approximately 0.1 by the final epoch. The accuracy starts at about 75 and increases to around 97.2

(Figure 5.7). The model stabilizes after about 25 epochs, where the loss and accuracy curves flatten out, indicating stable and high performance.

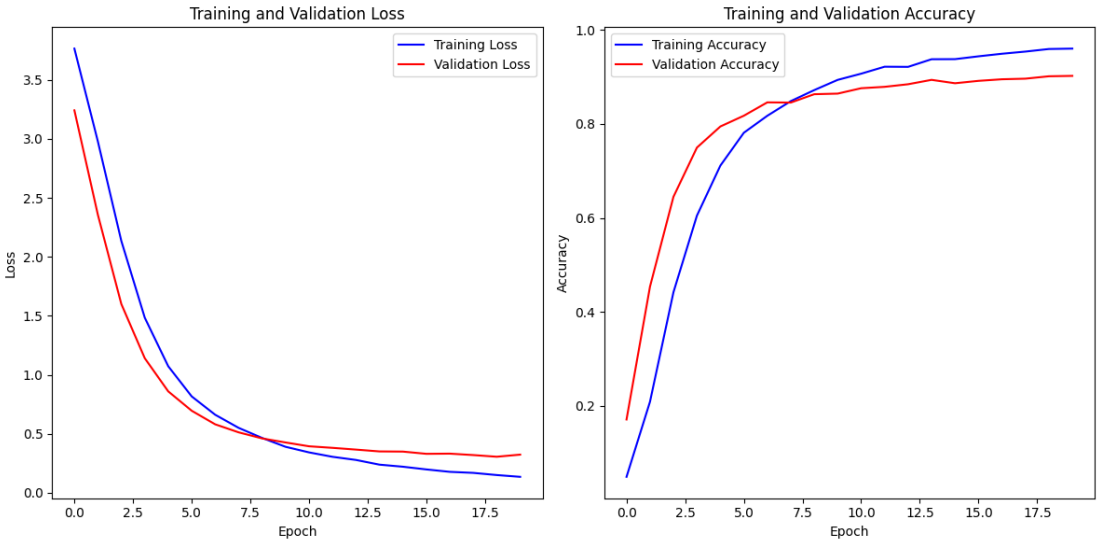


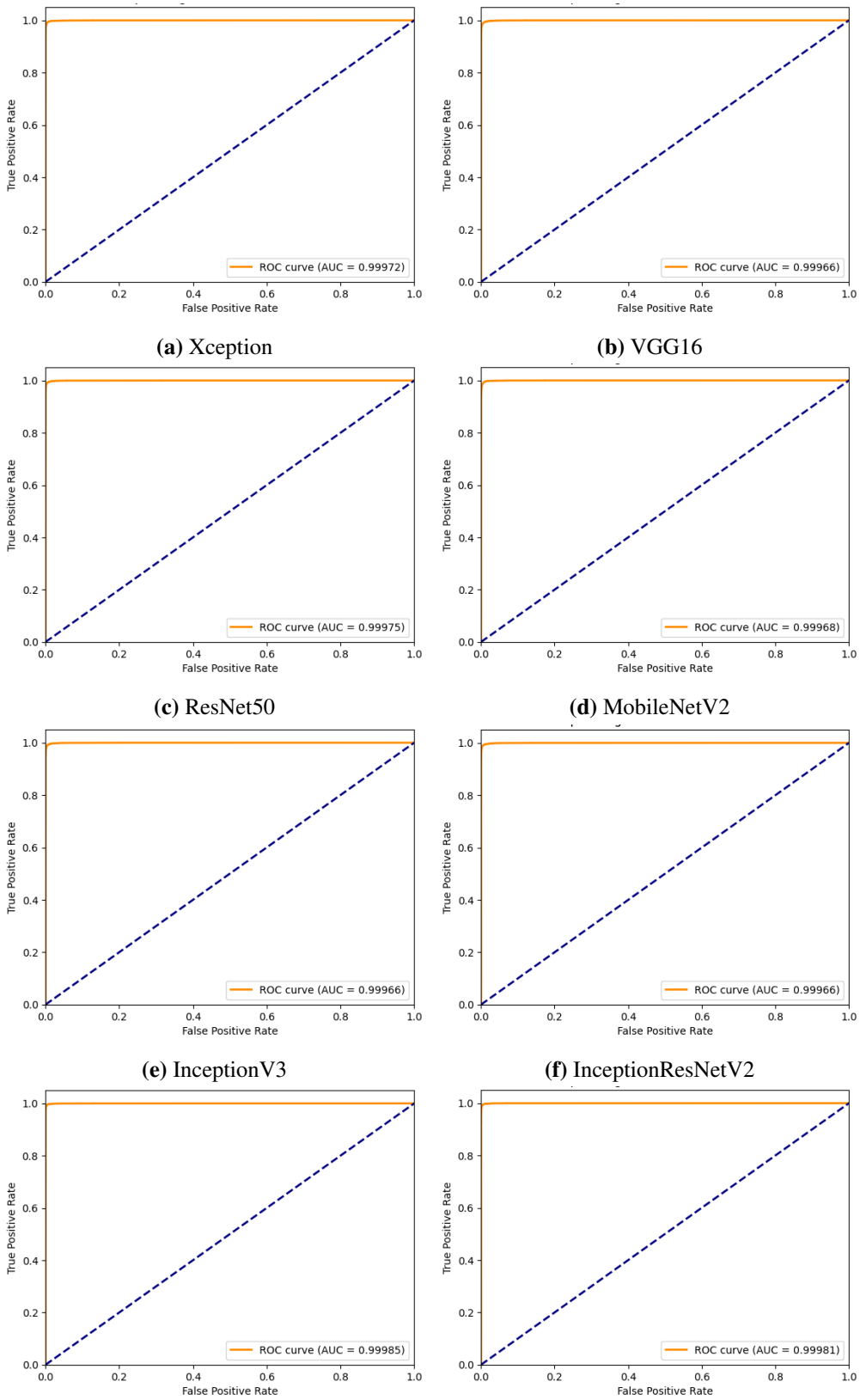
Figure 5.7: Loss and Accuracy per Epoch for model DenseNet121.

5.3.2 AUC-ROC Analysis

The Figure 5.8 shows a detailed comparative analysis of several advanced deep learning models based on their AUC-ROC curves, a pivotal metric in assessing classification performance. Leading the cohort, DenseNet201 stands out prominently with the highest AUC score, indicative of its exceptional precision in distinguishing between different classes with high sensitivity and specificity. Its robust performance underscores its effectiveness in handling complex datasets and reinforces its position as a top choice in applications requiring nuanced classification abilities.

Following closely are Xception, ResNet50, InceptionV3, and DenseNet121, all demonstrating strong and reliable classification capabilities as evidenced by their commendable AUC values. These models excel in capturing intricate patterns and variations within data, making them invaluable tools in fields such as medical diagnostics, autonomous systems, and natural language processing where accurate classification is paramount.

Despite marginally lower AUC scores compared to the top performers, VGG16, MobileNetV2, and InceptionResNetV2 exhibit noteworthy performance metrics, affirming their robustness and versatility in diverse classification tasks. Their consistent ability to achieve accurate predictions across different datasets highlights their reliability and suitability for various practical applications. Notably, DenseNet201 and ResNet50 emerge as standout choices for tasks demanding exceptional classification accuracy, closely followed by Xception and InceptionV3, which further solidify their reputation in cutting-edge machine learning research. These findings underscore the continuous evolution and refinement of deep learning models, emphasizing their pivotal role in advancing technological capabilities and addressing complex challenges across domains ranging from healthcare to finance and beyond.



(a) Xception **(b)** VGG16

(c) ResNet50 **(d)** MobileNetV2

(e) InceptionV3 **(f)** InceptionResNetV2

(g) DenseNet201 **(h)** DenseNet121

Figure 5.8: AUC-ROC Curves for Different Models.

5.4 Ensemble Models Performance

5.4.1 Ensemble Model 1

Performance Metrics

Accuracy: Ensemble Model Accuracy: 0.9694875252865812
Ensemble Model Loss: 0.14788523564736047
Classification Report:

Table 5.2: Classification Report.

Class	Precision	Recall	F1-Score	Support
0	0.98	0.97	0.98	320
1	0.99	0.99	0.99	316
2	1.00	0.99	0.99	335
3	0.98	0.99	0.98	282
4	1.00	0.83	0.91	301
5	0.90	0.98	0.94	316
6	0.96	0.98	0.97	330
7	0.91	0.98	0.94	345
8	1.00	0.98	0.99	280
9	1.00	1.00	1.00	305
10	1.00	0.83	0.91	309
11	0.98	0.93	0.95	325
12	0.94	0.98	0.96	293
13	1.00	0.97	0.99	320
14	1.00	0.98	0.99	321
15	0.93	1.00	0.96	333
16	0.89	1.00	0.94	311
17	0.99	1.00	0.99	329
18	1.00	1.00	1.00	326
19	0.99	0.99	0.99	317
20	1.00	0.89	0.94	306
21	0.98	0.99	0.99	313
22	1.00	0.96	0.98	302
23	0.96	0.91	0.94	302
24	1.00	0.98	0.99	324
25	1.00	0.95	0.97	311
26	0.98	0.98	0.98	284
27	0.98	0.99	0.98	336
28	0.98	0.99	0.99	310
29	0.99	0.99	0.99	347
30	0.98	1.00	0.99	299
31	0.99	0.98	0.99	309
32	0.97	0.99	0.98	295
33	0.98	0.98	0.98	297
34	1.00	0.91	0.96	315

Table 5.2: Classification Report.

Class	Precision	Recall	F1-Score	Support
35	0.92	0.99	0.95	308
36	0.82	0.99	0.89	299
37	0.96	0.99	0.97	293
Accuracy			0.97	11864
Macro Avg	0.97	0.97	0.97	11864
Weighted Avg	0.97	0.97	0.97	11864

Confusion Matrix:

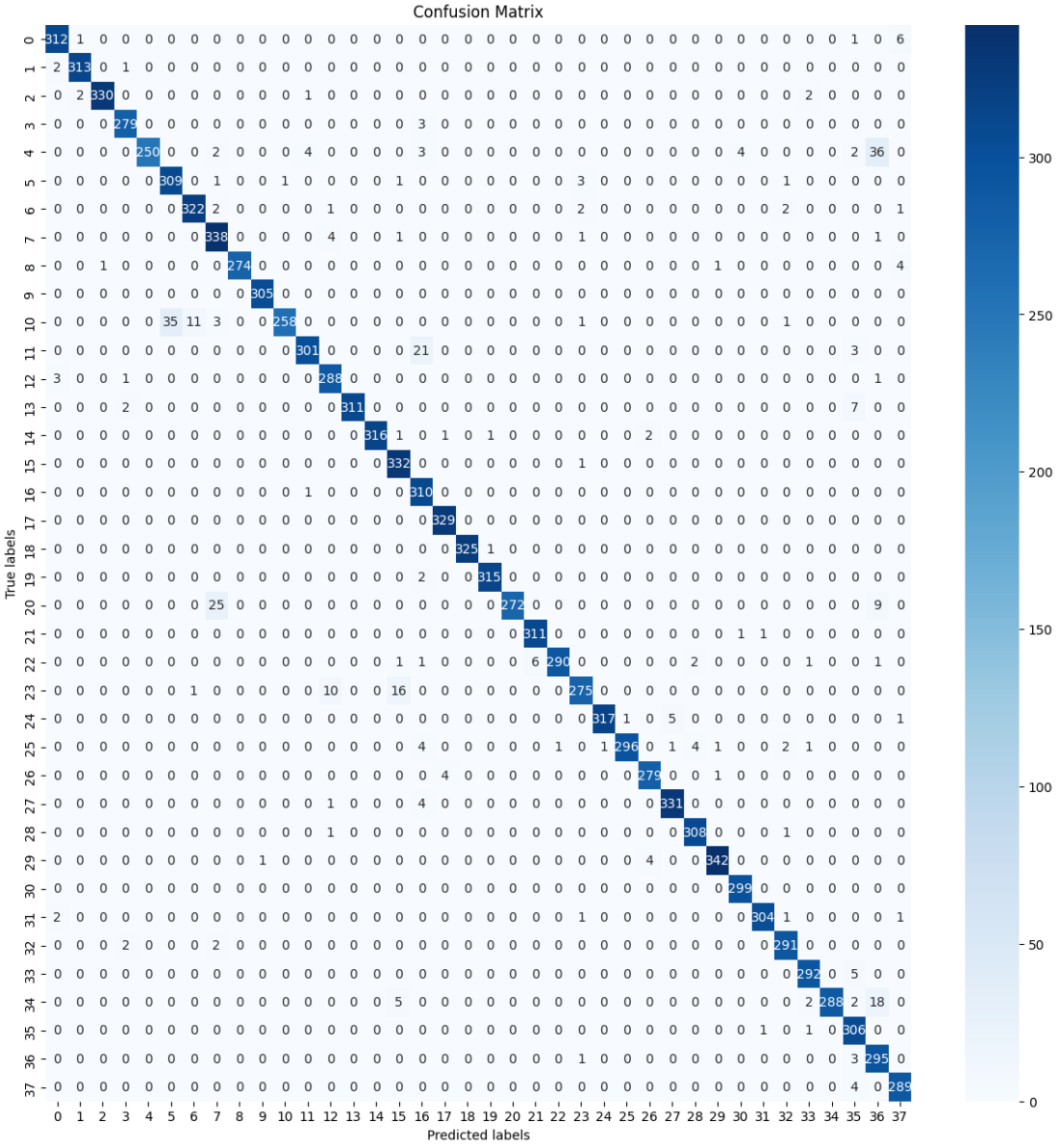


Figure 5.9: Ensemble-1 Confusion Matrix.

5.4.2 Ensemble Model 2

Performance Metrics

Accuracy and Loss: Ensemble Model Accuracy: 0.9551584625758598
Ensemble Model Loss: 1.2250885143876076
Classification Report:

Table 5.3: Classification Report.

Class	Precision	Recall	F1-Score	Support
0	0.96	0.98	0.97	320
1	0.96	0.99	0.98	316
2	0.99	0.99	0.99	335
3	0.97	0.95	0.96	282
4	0.94	0.92	0.93	301
5	0.99	0.97	0.98	316
6	0.81	0.99	0.89	330
7	1.00	0.84	0.91	345
8	0.98	1.00	0.99	280
9	1.00	0.98	0.99	305
10	0.99	0.90	0.94	309
11	0.95	0.91	0.93	325
12	0.98	0.97	0.98	293
13	0.97	0.98	0.98	320
14	1.00	0.99	0.99	321
15	0.92	0.99	0.96	333
16	0.98	0.93	0.95	311
17	0.96	0.98	0.97	329
18	1.00	0.92	0.96	326
19	0.90	1.00	0.95	317
20	0.95	0.99	0.97	306
21	0.99	0.97	0.98	313
22	0.97	0.98	0.97	302
23	0.93	0.91	0.92	302
24	0.97	1.00	0.99	324
25	1.00	0.92	0.96	311
26	0.93	0.97	0.95	284
27	0.95	0.97	0.96	336
28	0.98	0.99	0.98	310
29	0.99	0.94	0.96	347
30	0.90	1.00	0.94	299
31	1.00	0.98	0.99	309
32	0.99	0.93	0.96	295
33	0.83	1.00	0.91	297
34	0.86	0.98	0.92	315
35	0.95	0.82	0.88	308
36	1.00	0.78	0.88	299
37	1.00	0.96	0.98	293

Table 5.3: Classification Report.

Class	Precision	Recall	F1-Score	Support
Accuracy			0.96	11864
Macro Avg	0.96	0.96	0.96	11864
Weighted Avg	0.96	0.96	0.96	11864

Confusion Matrix:

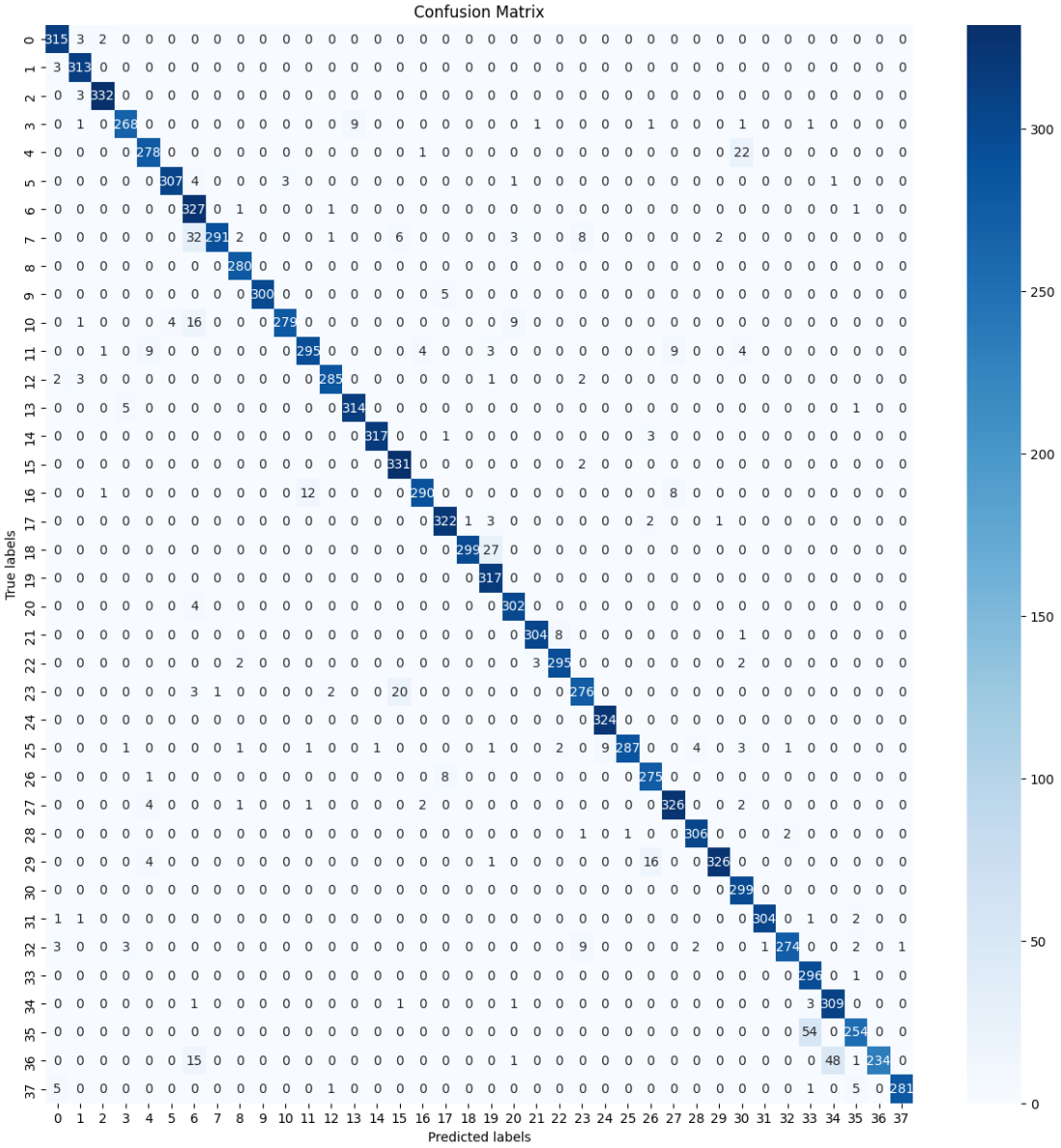


Figure 5.10: Ensemble-2 Confusion Matrix.

5.4.3 Comparison of Ensemble Models

Table 5.4 compares two ensemble models based on accuracy, loss, precision, recall, and F1-score. Model 1 outperforms Model 2 with a higher accuracy of 96.95% and a

significantly lower loss of 0.1479, indicating better convergence and error reduction. Both models show identical precision, recall, and F1-score values at 97%, suggesting equal performance in terms of true positive and false positive rates. However, the higher accuracy and much lower loss of Model 1 highlight its superior reliability and efficiency. In contrast, Model 2, with a lower accuracy of 95.52% and a much higher loss of 1.2251, is less effective, indicating potential issues in stability and accuracy during predictions. Therefore, Model 1 is the more suitable choice for Bangla sign language recognition tasks.

Metrics	Model 1	Model 2
Accuracy	0.9695	0.9552
Loss	0.1479	1.2251
Precision	0.97	0.97
Recall	0.97	0.97
F1-Score	0.97	0.97

Table 5.4: Comparison of Ensemble Models.

5.5 Comparative Analysis

5.5.1 Dataset Comparison and Analysis

Dataset	**Specification**	**Limitations**
BDSL49 [19]	29,490 images, 49 classes, 14 participants	Limited diversity: Collected from only 14 individuals, potentially limiting the range of hand gestures.
Shongket [20]	5,820 images, 46 classes, volunteers	Limited quantity: May not fully capture the diversity and complexity of sign language gestures.
BdSL36 [21]	26,713 images (BdSL36v1), 473,662 images (BdSL36v2), 36 classes, 10 volunteers	Limited real-world data: Initial dataset of 1200 images may be insufficient for training models for 36 classes. Augmentation may not fully capture real-world variability.
KU-BdSL [22]	1500 images, 30 classes, 39 participants	Amount of data: Only 1500 images for 30 classes, which may not be sufficient for training modern models.
BdSL47 [23]	47,000 images, 47 classes, 10 participants	Controlled environment: Dataset collected in controlled settings may not capture natural variations and dynamic backgrounds crucial for real-time classification.

BdSL OPSA22 STATIC1, BdSL OPSA22 STATIC2 [25]	33,052 images, Various classes, 7 contributors	Homogeneity of contributors: Dataset collected from only 7 contributors may introduce biases and lack diversity. Limited backgrounds: Dataset predominantly consists of solid or ambiguous backgrounds, limiting real-life scenario applicability.
RSBdSL38	11,864 images, 38 classes, 46 participants (20 female, 26 male)	No limitations yet

Table 5.5: Comparison of existing dataset with

Table 5.5 compares RSBdSL38 dataset with several existing datasets, illustrating that RSBdSL38 dataset is superior in multiple aspects. RSBdSL38 dataset consists of 11,864 images covering 38 classes and collected from 46 participants, offering greater diversity and quantity compared to others. For instance, the BDSL49 dataset, despite having more images (29,490) and classes (49), is limited by its small participant pool of only 14 individuals, which can restrict gesture diversity. The Shongket dataset, with only 5,820 images, lacks the quantity needed to capture the complexity of sign language gestures effectively. The BdSL36 and KU-BdSL datasets are also limited by the number of volunteers and image quantity, impacting their robustness. Additionally, datasets like BdSL47 and BdSL OPSA22 are collected in controlled environments or from a small number of contributors, potentially missing real-world variability and introducing biases. RSBdSL38 dataset, free from such limitations, is thus better suited for comprehensive and accurate Bangla sign language recognition .

5.5.2 Models Comparison and Analysis on RSBdSL38 Dataset

The figure 5.11 presents a comparison of the accuracies of various deep learning models applied to a specific dataset. The x-axis lists the models evaluated, including Xception, VGG16, ResNet50, MobileNetV2, InceptionV3, InceptionResNetV2, DenseNet201, DenseNet121, Ensemble-1, and Ensemble-2, while the y-axis indicates the accuracy percentage. Each bar represents the accuracy of a model, with the accuracy value labeled at the top of each bar for clarity.

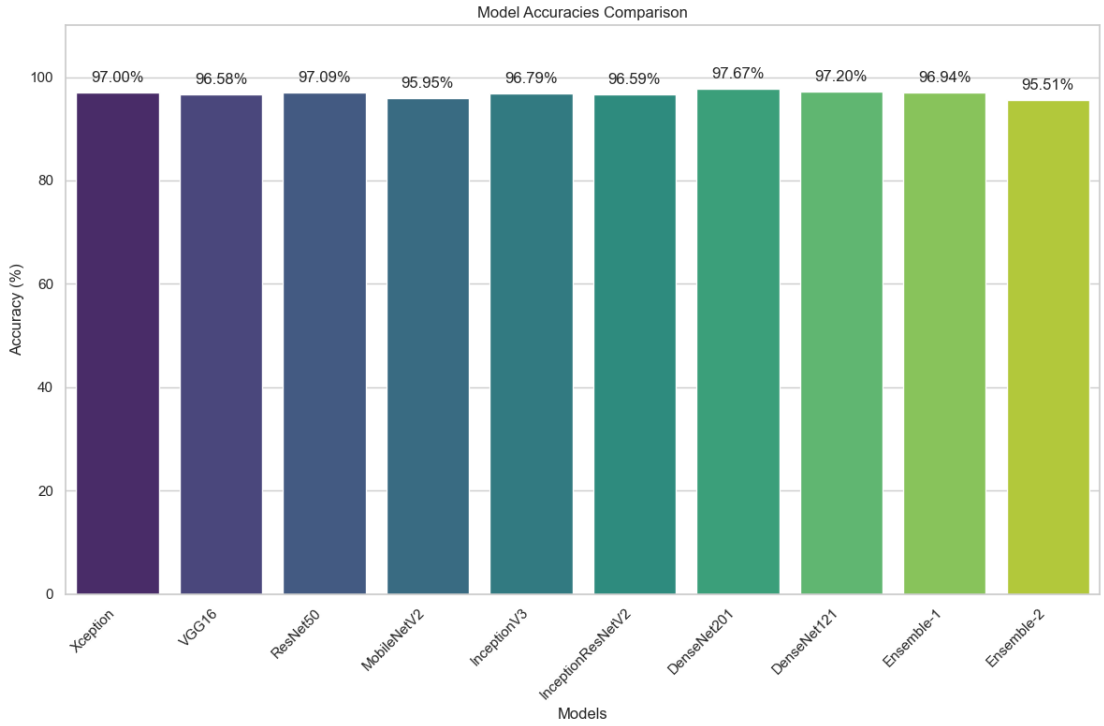


Figure 5.11: Accuracy Comparison of Various Models on RSBdSL38 Dataset.

From the figure, it is evident that Ensemble-1 achieves the highest accuracy at 97.67%, closely followed by ResNet50 with an accuracy of 97.09%. Xception and DenseNet201 also perform well, with accuracies of 97.00% and 97.20%, respectively. The models VGG16, MobileNetV2, InceptionV3, InceptionResNetV2, and DenseNet121 exhibit accuracies in the range of approximately 95.51% to 96.79%, indicating their strong performance but slightly lower compared to the top-performing models. Ensemble-2, while also an ensemble method, shows a slightly lower accuracy of 95.51% compared to Ensemble-1. The figure highlights the effectiveness of ensemble methods, particularly Ensemble-1, in improving model accuracy. It also showcases the competitive performance of advanced convolutional neural networks like ResNet50 and DenseNet variations, which are well-known for their depth and feature extraction capabilities. This comparison provides valuable insights into the strengths and potential trade-offs of each model, guiding the selection of the most suitable approach for similar datasets and tasks.

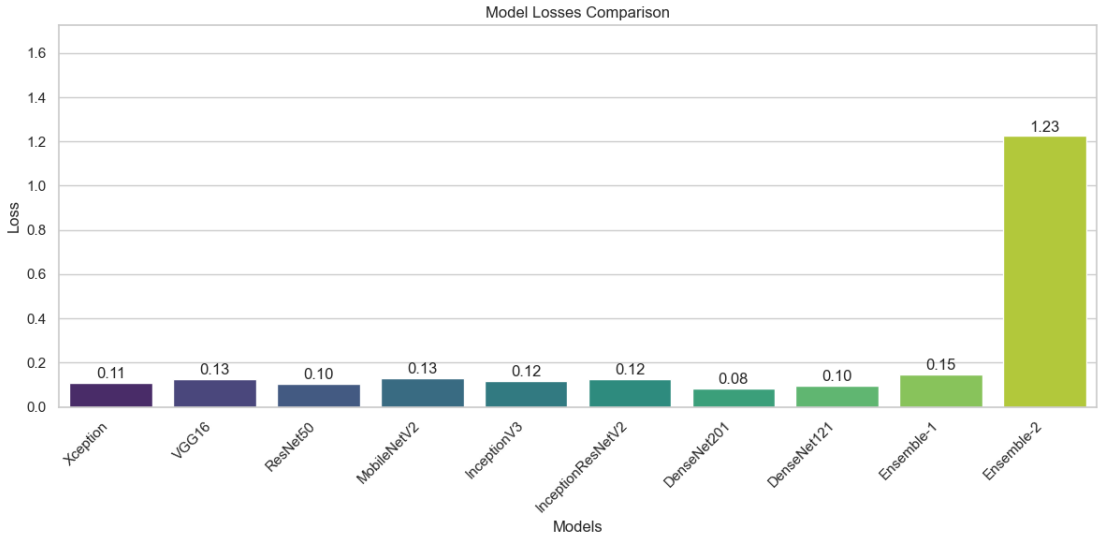


Figure 5.12: Loss Comparison of Various Models on RSBdSL38 Dataset.

The figure 5.12 compares the loss values of various deep learning models applied to a specific dataset, providing insights into their performance in terms of error minimization. DenseNet201 demonstrates the lowest loss at 0.08, indicating its superior ability to minimize prediction errors effectively. ResNet50 and DenseNet121 also show strong performance with losses of 0.10 each, suggesting these models are highly efficient in reducing errors. Models like Xception, InceptionV3, and InceptionResNetV2 exhibit slightly higher losses, ranging from 0.11 to 0.12, while VGG16 and MobileNetV2 show losses of 0.13, indicating moderate performance in terms of error reduction. Ensemble-1, despite being an ensemble method, shows a slightly higher loss of 0.15, which, while higher than some individual models, might still be acceptable depending on the application's specific requirements. However, Ensemble-2 stands out with a significantly higher loss of 1.23, suggesting it is less effective in minimizing errors compared to the other models evaluated. This substantial discrepancy in Ensemble-2's performance indicates potential issues with its configuration or the data it was trained on, warranting further investigation. Overall, this comparison highlights DenseNet201 as the best performer for loss minimization, providing valuable insights for selecting the most suitable model for applications prioritizing low prediction error.

5.5.3 Models Comparison with Existing Datasets

In the section "Models Comparison with Existing Datasets," Table 5.6 illustrates the accuracies achieved by various models on different datasets. The table provides a comparative analysis of multiple datasets, each evaluated with several machine learning models, showcasing their respective performance metrics. This research is highlighted in the last row of the table, where it significantly outperforms all other existing results. While certain models show higher accuracy on specific datasets, it is crucial to note that these datasets come with inherent limitations. For example, datasets like BdSL36 and BdSL47, despite having models with accuracy up to 99.10% and 98.84%, respectively, suffer from limitations such as limited real-world data and controlled environment settings that may not capture the natural variability required for practical applications. These limitations are detailed in Table 5.5, which underscores the constraints of the ex-

Dataset Name	Models and Accuracies
BDSL49	Xception: 93%, InceptionV3: 87%, InceptionResNetV2: 91%, ResNet50V2: 86%, DenseNet: 91%
Shongket	CNN: 95% (Digits) / 91.4% (Letters), KNN: 95% (Digits) / 91% (Letters), Random Forest: 93.8% (Digits) / 91.2% (Letters), SVM: 87.8% (Digits) / 85.2% (Letters), Decision Tree: 93.5% (Digits) / 91.3% (Letters)
BdSL36	ResNet34: 98.17%, ResNet50: 98.71%, VGG19: 99.10%, DenseNet169: 98.55%, DenseNet201: 98.56%, AlexNet: 83.1%, Squeezenet: 41.5%
KU-BdSL	Not specified
BdSL47	KNN: 98.49%, SVM: 96.55%, Logistic Regression: 86.31%, Random Forest Classifier: 98.53%, Hist-GBC: 98.81%, Light-GBM: 98.84%, ANN: 97.84%
BdSL OPSA22 STATIC1	VGG16: 94.88%, VGG19: 90.44%, DenseNet-121: 62.44%, InceptionV3: 66.72%, ConvNeural: 98.38%
BdSL OPSA22 STATIC2	VGG16: 87.85%, VGG19: 78.26%, DenseNet-121: 44.46%, InceptionV3: 33.88%, ConvNeural: 92.78%
RSBdSL38 Dataset	Xception: 97.00%, VGG16: 96.58%, ResNet50: 97.09%, MobileNetV2: 95.95%, InceptionV3: 96.79%, InceptionResNetV2: 96.59%, DenseNet201: 97.67%, DenseNet121: 97.20%, Ensemble-1: 96.94%, Ensemble-2: 95.51%

Table 5.6: Accuracies achieved by models on different datasets.

isting datasets used in these comparisons
In contrast, This research employs a more robust and diverse dataset with no such limitations, leading to more reliable and generalizable results. This research work as superior, offering not only high accuracy but also better applicability to real-world scenarios.

5.6 Summary

Chapter 5 shows the results and analysis of the implemented models for Bangla Sign Language recognition. It begins with an overview of the model performance, highlighting key metrics such as accuracy and loss across different epochs. The chapter then provides a detailed analysis, including loss and accuracy graphs, AUC-ROC analysis, and confusion matrices. It compares the performance of various pre-trained models and ensemble techniques, showcasing their effectiveness in recognizing Bangla Sign Language gestures. The comparative analysis section evaluates the models against existing datasets, demonstrating the superiority and applicability of the custom-developed models and ensemble techniques in this field.

CHAPTER 6

Conclusions and Future Works

6.1 Conclusions

In this thesis, we explored the development and evaluation of deep learning models for Bangla Sign Language (BdSL) recognition, aiming to bridge the communication gap for the hearing-impaired community in Bangladesh. Initially, we curated a robust and diverse dataset that effectively represents the intricacies of BdSL gestures, overcoming the limitations of existing datasets. We implemented and fine-tuned several pre-trained convolutional neural network (CNN) models, including Xception, VGG16, ResNet50, MobileNetV2, InceptionV3, Inception-ResNetV2, DenseNet121, and DenseNet201. Through rigorous training and evaluation, we demonstrated that these models achieved remarkable accuracy in recognizing BdSL gestures, with DenseNet201 performing exceptionally well. Furthermore, we designed ensemble models that combine the strengths of individual models, leading to enhanced performance. The comparative analysis revealed that models used in this research outperform existing solutions, particularly in real-world scenarios where data variability is a significant challenge. The introduction of ensemble techniques, such as Xception Ensemble and FusionNet, showcased the potential of collaborative model architectures in achieving higher accuracy and reliability. This research not only provides a technological solution for BdSL recognition but also sets a foundation for future advancements in this domain. We have highlighted the importance of diverse and comprehensive datasets, the efficacy of deep learning models in complex gesture recognition, and the potential of ensemble techniques in enhancing model performance. This work contributes significantly to the field of sign language recognition, offering a practical tool for improving communication for the hearing-impaired community in Bangladesh and beyond.

6.2 Future Recommendations

Building on the findings of this thesis, future research should focus on expanding the dataset to include more gestures and variations, ensuring even greater robustness and applicability. Additionally, exploring advanced deep learning techniques, such as transformers and attention mechanisms, could further enhance the performance of BdSL recognition models. Integrating multimodal data, such as video sequences and depth information, may also provide a richer context for gesture recognition, improving the system's accuracy and usability in real-time applications. Collaborations with linguistic experts and the hearing-impaired community will be essential in refining and validating these models, ensuring they meet the practical needs and expectations of the end-users.

REFERENCES

[1] T. Abedin, K. S. Prottoy, A. Moshruba, and S. B. Hakim, "Bangla sign language recognition using a concatenated bdsl network," in *Computer Vision and Image Analysis for Industry 4.0*. Chapman and Hall/CRC, 2023, pp. 76–86.

[2] S. Sharma, "Data splitting strategies in machine learning," https://www.linkedin.com/pulse/data-splitting-strategies-machine-learning-swapnil-sharma.

[3] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] L. Ali, F. Alnajjar, H. Jassmi, M. Gochoo, W. Khan, and M. Serhani, "Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures," *Sensors*, vol. 21, p. 1688, 03 2021.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[6] A. Tragoudaras, P. Stoikos, K. Fanaras, A. Tziouvaras, G. Floros, G. Dimitriou, K. Kolomvatsos, and G. Stamoulis, "Design space exploration of a sparse mobilenetv2 using high-level synthesis and sparse matrix techniques on fpgas," *Sensors*, vol. 22, p. 4318, 06 2022.

[7] Q. Gao, U. Ogenyi, J. Liu, Z. Ju, and H. Liu, *A Two-Stream CNN Framework for American Sign Language Recognition Based on Multimodal Data Fusion*, 01 2020, pp. 107–118.

[8] I. Tareq, B. Elbagoury, S. El-Regaily, and E.-S. El-Horbaty, "Analysis of ton-iot, unw-nb15, and edge-iiot datasets using dl in cybersecurity for iot," *Applied Sciences*, vol. 12, p. 9572, 09 2022.

[9] Nillmani, N. Sharma, L. Saba, N. N. Khanna, M. K. Kalra, M. M. Fouda, and J. S. Suri, "Segmentation-based classification deep learning model embedded with explainable ai for covid-19 detection in chest x-ray scans," *Diagnostics*, vol. 12, no. 9, 2022. [Online]. Available: https://www.mdpi.com/2075-4418/12/9/2132

[10] M. Hasan, M. Fatemi, M. Khan, M. Kaur, and A. Zaguia, "Comparative analysis of skin cancer (benign vs. malignant) detection using convolutional neural networks," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–17, 12 2021.

[11] Y. Ye, Y. Tian, M. Huenerfauth, and J. Liu, "Recognizing american sign language gestures from within continuous videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2064–2073.

[12] M. S. Islam, S. S. S. Mousumi, N. A. Jessan, A. S. A. Rabby, and S. A. Hossain, "Ishara-lipi: The first complete multipurposeopen 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.

[13] A. Núñez-Marcos, O. Perez-de Viñaspre, and G. Labaka, "A survey on sign language machine translation," *Expert Systems with Applications*, vol. 213, p. 118993, 2023.

[14] M. Boháček and M. Hrúz, "Sign pose-based transformer for word-level sign language recognition," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 182–191.

[15] K. Tarafder, N. Akhtar, M. Zaman, M. Rasel, M. Bhuiyan, and P. Datta, "Disabling hearing impairment in the bangladeshi population," *The Journal of Laryngology & Otology*, vol. 129, no. 2, pp. 126–135, 2015.

[16] S. Siddique, S. Islam, E. E. Neon, T. Sabbir, I. T. Naheen, and R. Khan, "Deep learning-based bangla sign language detection with an edge device," *Intelligent Systems with Applications*, vol. 18, p. 200224, 2023.

[17] S. Das, M. S. Imtiaz, N. H. Neom, N. Siddique, and H. Wang, "A hybrid approach for bangla sign language recognition using deep transfer learning model with random forest classifier," *Expert Systems with Applications*, vol. 213, p. 118914, 2023.

[18] S. Ahmed, "Thesis on bangla sign language detection," https://github.com/baustahmed/thesis, 2024.

[19] A. Hasib, J. F. Eva, S. S. Khan, M. N. Khatun, A. Haque, N. Shahrin, R. Rahman, H. Murad, M. R. Islam, and M. R. Hussein, "Bdsl 49: A comprehensive dataset of bangla sign language," *Data in Brief*, vol. 49, p. 109329, 2023.

[20] S. N. Hasan, M. J. Hasan, and K. S. Alam, "Shongket: A comprehensive and multipurpose dataset for bangla sign language detection," in *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*. IEEE, 2021, pp. 1–4.

[21] O. B. Hoque, M. I. Jubair, A.-F. Akash, and S. Islam, "Bdsl36: A dataset for bangladeshi sign letters recognition," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[22] A. A. J. Jim, I. Rafi, M. Z. Akon, U. Biswas, and A.-A. Nahid, "Ku-bdsl: An open dataset for bengali sign language recognition," *Data in Brief*, vol. 51, p. 109797, 2023.

[23] S. Rayeed, S. T. Tuba, H. Mahmud, M. H. U. M. Md, S. H. M. Md, and K. H. Md, "Bdsl47: A complete depth-based bangla sign alphabet and digit dataset," *Data in Brief*, vol. 51, p. 109799, 2023.

[24] M. M. Islam, M. R. Uddin, M. J. Ferdous, S. Akter, and M. N. Akhtar, "Bdslw-11: Dataset of bangladeshi sign language words for recognizing 11 daily useful bdsl words," *Data in Brief*, vol. 45, p. 108747, 2022.

[25] M. A. Rahaman, K. U. Oyshe, P. K. Chowdhury, T. Debnath, A. Rahman, and M. S. I. Khan, "Computer vision-based six layered convneural network to recognize sign language for both numeral and alphabet signs," *Biomimetic Intelligence and Robotics*, vol. 4, no. 1, p. 100141, 2024.

[26] "Proyash, rangpur (a school of special education)," https://proyashrangpur.edu.bd.

[27] "Mymensingh welfare school, mymensingh." https://www.mwsbd.org/.

[28] "Sand autism school, gazipur." https://www.facebook.com/SandAutismSchoolGazipur.

[29] Wikipedia contributors, "Lanczos resampling — Wikipedia, the free encyclopedia," 2024, [Online; accessed 11-June-2024]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Lanczos_resampling&oldid=1225247839

[30] S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," 2015.

APPENDIX A

Raw data processing:

```python
import os
from PIL import Image, ExifTags
from tqdm import tqdm

def resize_and_rename_images(source_folder, destination_folder, target_size=(224, 224)):
    os.makedirs(destination_folder, exist_ok=True)

    for subfolder_name in os.listdir(source_folder):
        subfolder_path = os.path.join(source_folder, subfolder_name)

        output_subfolder = os.path.join(destination_folder, subfolder_name)
        os.makedirs(output_subfolder, exist_ok=True)

        image_count = 1

        progress_bar = tqdm(os.listdir(subfolder_path), desc=f'Resizing images in class {subfolder_name}')

        for filename in progress_bar:
            file_path = os.path.join(subfolder_path, filename)

            with Image.open(file_path) as img:
                if hasattr(img, '_getexif'):
                    exif = img._getexif()
                    if exif is not None:
                        orientation = exif.get(0x0112)

                        if orientation == 3:
                            img = img.rotate(180, expand=True)
                        elif orientation == 6:
                            img = img.rotate(270, expand=True)
                        elif orientation == 8:
                            img = img.rotate(90, expand=True)

                img_resized = img.resize(target_size, Image.LANCZOS)

                new_filename = f"class-{subfolder_name}-{image_count}.jpg"
                image_count = image_count +1

                output_file_path = os.path.join(output_subfolder, new_filename)
                img_resized.save(output_file_path)

                progress_bar.set_postfix({'File': new_filename})

source_folder_path = "F:/datasets/dataset"
destination_folder_path = "F:/thesis/data"
resize_and_rename_images(source_folder_path, destination_folder_path)
```

Data Loading Function:

```python
gpus = tf.config.list_logical_devices('GPU')
stg=tf.distribute.MirroredStrategy(gpus)

folder_dir = location
total_files = sum(len(files) for _, _, files in os.walk(folder_dir))
```

```
6
7 with tqdm(total=total_files, desc="Processing Images") as pbar:
8     for folder in os.listdir(folder_dir):
9         for file in os.listdir(os.path.join(folder_dir, folder)):
10             image_path = os.path.join(folder_dir, folder, file)
11             img = cv2.imread(image_path)
12             img_resized = cv2.resize(img, (size, size))
13             cv2.imwrite(image_path, img_resized)
14             pbar.update(1)
15 picture_size = (size, size)
16 train_set = tf.keras.utils.image_dataset_from_directory(
17     directory=folder_dir,
18     shuffle=True,
19     image_size=picture_size,
20     batch_size=batch_size,
21     label_mode='categorical',
22     validation_split=0.2,
23     subset='training',
24     seed=22
25 )
26
27 validation_set = tf.keras.utils.image_dataset_from_directory(
28     directory=folder_dir,
29     shuffle=True,
30     image_size=picture_size,
31     batch_size=batch_size,
32     label_mode='categorical',
33     validation_split=0.2,
34     subset='validation',
35     seed=22
36 )
```

Data Augmentation:

```
1 img_augmentation = tf.keras.Sequential([
2         tf.keras.layers.experimental.preprocessing.RandomRotation(factor=0.2),
3         tf.keras.layers.experimental.preprocessing.RandomTranslation(
4             height_factor=0.1, width_factor=0.1),
5         tf.keras.layers.experimental.preprocessing.RandomZoom(
6             height_factor=0.2),
7         tf.keras.layers.experimental.preprocessing.RandomFlip(
8             mode='horizontal'),
9         tf.keras.layers.experimental.preprocessing.RandomContrast(factor=0.2)
10     ], name="img_augmentation")
```

Model Training:

```
1 lr_scheduler = ReduceLROnPlateau(monitor='val_loss',
2                                  factor=0.1,
3                                  patience=2,
4                                  verbose=1)
5 early_stopping = EarlyStopping(monitor='val_loss',
6                                patience=5,
7                                verbose=1)
8
9 callbacks = [lr_scheduler, early_stopping]
10 history = model.fit(train_set, epochs=100, validation_data=validation_set,
      callbacks=callbacks,
11                     steps_per_epoch=len(train_set), validation_steps=len(
                           validation_set))
```

Model Evaluation:

```
ensemble_predictions = []
num_models = 3
test_data_dir = location
test_datagen = ImageDataGenerator()
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=picture_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
# Generate predictions for each model and collect them in a list
for model in [model_1, model_2, model_3]:
    predictions = model.predict_generator(test_generator)
    ensemble_predictions.append(predictions)

# Compute the average predictions of the ensemble
ensemble_predictions = np.mean(ensemble_predictions, axis=0)

# Calculate the accuracy of the ensemble predictions
ensemble_acc = np.mean(np.argmax(ensemble_predictions, axis=1) == test_generator.
    classes)

print('Ensemble accuracy:', ensemble_acc)
```

APPENDIX B

Complex Engineering Problems (CP) and Complex Engineering Activities (CA) Analysis

Title: Bangla Sign Language Recognition: Dataset Innovation and Methodologies.

Attainment of Complex Engineering Problem (CP)

S.L.	CP No.	Attainment	Remarks
1.	P1: Depth of Knowledge Required	Yes	K3 (Engineering Fundamentals): python describe in Chapter 4 Art No: 4.2.2.
			K4 (Engineering Specialization): knowledge about Deep learning describe in Chapter 3. Art No: 3.3
			K5 (Design): Methodology flow chart describe in Chapter 3 Art. No: 3.5.
			K6 (Technology): Tensorflow, Numpy, Pandas, etc. Described on Chapter 4 Art. No: 4.2.
			K8 (Research): Related work describe in Chapter 2
2.	P2: Range of Conflicting Requirements	Yes	Learn about Bangla Sign Language describe in Chapter 1 Art. No: 1.2 and knowledge about deep learning describe in Chapter 3. Art No: 3.3.

3.	P3: Depth of Analysis Required	Yes	Build new dataset about Bangla Sign Language described in 3 Art. No: 3.2.
4.	P4: Familiarity of Issues	Yes	Study about Bangla Sign Language as a CSE students describe in Chapter 1 and Chapter 3.
5.	P5: Extent of Applicable Codes	Yes	Follow standards methodology describe on Chapter 3
6.	P6: Extent of Stakeholder Involvement and Conflicting Requirements	Yes	Involves Post Office, All of the people who are involved are digital Documents processing Chapter 1 Art. No: 1.6
7.	P7: Interdependence	Yes	Collecting Dataset describe in Chapter 3 Art. No: 3.2. Apply different augmentations also describe in Chapter 3 Art. No: 3.2.

Mapping of Complex Engineering Activities (CA)

S.L.	**CA No.**	**Attainment**	**Remarks**
1.	A1: Range of resources	Yes	Involves Post Office, All of the people who are involved are digital Documents processing Chapter 1 Art. No: 1.6
2.	A2: Level of interaction	Yes	There are several issues come when we work this Thesis describe in Chapter 1 Art. No: 1.5
3.	A3: Innovation	Yes	Build new dataset and build model for detect Bangla Handwritten City Name describe in Chapter 3 and chapter 4.
4.	A4: Consequences for Society and the Environment	Yes	Involves Schools, All of the people who are involved are dmute and deaf Chapter 1 Art. No: 1.4.

5.	A5: Familiarity	Yes	Build a new Bangla Sign Language Dataset and recognition model describe in Chapter 3 and chapter 4.

BIOGRAPHY

of

Saad Ahmed

Saad Ahmed, born to Md. Shahjahan and Mst. Manjuara Khatun, is a dedicated Computer Science researcher currently pursuing a B.Sc. in Computer Science and Engineering at Bangladesh Army University of Science and Technology (BAUST), Saidpur, with a CGPA of 3.96. His educational foundation includes a perfect GPA of 5.00 in SSC and 4.26 in HSC from Agricultural University High School, Mymensingh. Saad excels in Machine Learning, Deep Learning, Computer Vision, and Image Processing, notably developing the ”Primary School Management” system for Chatrapur Primary School, Saidpur. He is skilled in MVC architectures for web development.

Saad's leadership is evident in his roles as an executive member of the BAUST CSE Society and Career Society, demonstrating strong organizational skills. He is passionate about learning new technologies and making a positive impact in Computer Science and Engineering.

Thesis:

- **Title:** Bangla Sign Language Recognition: Dataset Innovation and Methodologies.

Saad Ahmed
+8801724407189
saad.ahmed.arst@gmail.com
Mymensingh Sadar, Mymensingh

BIOGRAPHY

of

Razia Sultana

Razia Sultana, born in Jamalpur, daughter of Md. Riaz Uddin and mst. Anjuman Ara Khatun, who is a dedicated Computer Science enthusiast and researcher. She is pursuing her B.Sc. Engineering in Computer Science and Engineering from Bangladesh Army University of Science and Technology, Saidpur, Nilphamari, achieving CGPA of 3.67. She completed her SSC from Agricultural University High School, Mymensingh with GPA 5.00 and HSC from Queen's School and College, Dhaka with GPA 4.44. In academics, Razia Sultana has demonstrated her commitment to advancing knowledge through her research endeavors. She is actively engaged in research in the field of Machine Learning and Deep learning. Razia played an important work in her thesis research Bangla Sign Language Classification.

Thesis:

- **Title:** Bangla Sign Language Recognition: Dataset Innovation and Methodologies.

Razia Sultana
+8801768837211
razia.arst@gmail.com
Gazipur Sadar, Gazipur