Tri-Modal Ensemble for Enhanced Bangla Sign Language Recognition

by

Khan Abrar Shams
19201052
Md. Rafid Reaz
19201044
Sanjida Islam
20101615
Mohammad Ryan Ur Rafi
22241152

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
September 2023

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Khan Abrar Shams

Khan Abrar Shams
19201052

Md. Rafid Reaz

Md. Rafid Reaz
19201044

Sanjida Islam

Sanjida Islam
20101615

Mohammad Ryan Ur Rafi
22241152

Approval

The thesis/project titled "Tri-Modal Ensemble for Enhanced Bangla Sign Language Recognition" submitted by

1. Khan Abrar Shams (19201052)
2. Md Rafid Reaz (19201044)
3. Sanjida Islam (20101615)
4. Mohammad Ryan Ur Rafi (22241152)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 17, 2023.

Examining Committee:

Supervisor:
(Member)

Md. Shahriar Rahman
Lecturer
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Mr. Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

Sign language is the most common method of communication for people with disabling hearing loss. Bangladesh, where BdSL is prominently used among the disabling people, finds communicating with the general mass challenging. Thus, a system to understand BdSL accurately and efficiently has become a popular demand. Deep learning architectures such as CNN, ANN, RNN, and Axis Independent LSTM can interpret Bangla Sign Language into readable digital wording. Commonly, an image-based sign language recognition system contains a recording camera that continuously sends images to a model. The model then provides a prediction based on those images. However, it creates a lot of uncertainty variables, such as the lighting issue, noisy background, skin color, and hand orientations. To this end, we propose a procedure that can reduce this uncertainty variable by considering three different modalities, spatial information, skeleton awareness, and edge awareness. We propose three image pre-processing techniques and integrate three convolutional neural network models. Finally, we tested out nine different ensemble meta-learning algorithms where five of the algorithms are modifications of averaging and voting techniques. As a result, our proposed model achieved higher training accuracy at 99.77%, 98.11%, and 99.30% than any other state-of-the-art image classification architectures except for ResNet50 at 99.87%. We achieved the highest accuracy of 95.13% on the testing set. This research shows that considering multiple modalities can improve the system's overall performance.

Keywords: Bangla Sign Language (BdSL) · Convolutional Neural Network · Ensemble Method

Acknowledgement

First and foremost, all praise to the Great Allah for whom our pre-thesis-1 has been completed without little to no interruption. We are very grateful to our supervisor Md. Shahriar Rahman and co-supervisor Mr Rafeed Rahman, for their invaluable and tremendous support, advice, and patience during our pre-thesis-1. And last but not least, we would like to thank our friends and family for their encouragement throughout the journey. The accomplishment does not just belong to us, And it would not have been possible if it weren't for these very important people in our life

Table of Contents

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN Convolutional Neural Network

$LSTM$ Long Short-Term Memory

ORB Oriented FAST and Rotated BRIEF

RNN Recurrent Neural Network

$SAM - SLR$ Skeleton Aware Multi-modal Sign Language Recognition

$SL - GCN$ Sign Language Graph Convolution Network

SLR Sign Language Recognition

$SSTCN$ Separable Spatial-Temporal Convolution Network

SVM Support Vector Machines

Chapter 1

Introduction

1.1 Problem Statement

A study says that by the year 2050, more than 700 million people – or 8.87% of people [21] will have disabling hearing loss. The World Health Organization (WHO) estimates that 430,000,000 people, or 1 in every 20 people worldwide, are diagnosed with disabling hearing loss [50]. It means more than 5% of the world suffers from hearing impairment. Sign language is vital in helping, assisting, and creating a connection between people with normal hearing and people from the deaf community. A 2013 study shows 32 million out of 360 million people suffering from hard of hearing are children [47]. In Bangladesh, disabling hearing loss is almost double, which stands at around 9.6% [12]. BdSL, or Bangla Sign Language, is one of the major sign languages used in Bangladesh. BdSL and WBSL are similar lexically but used prominently in their respective regions [17]. BdSL is a full-board lingua with identical Bengali-type symbols or alphabets that match numerous linguistic similarities via spoken linguistics. Even though learning sign language is essential, it is arduous for a person with normal hearing to master sign language quickly. So a system to interpret Bangla sign language is in dire need. This research aims to fulfill the need for interpreting Bangla sign language and provide an extensive guidelines to enhance the system and make it more accurate.

Supervised learning can be broken down into classification and regression problems. Classification, or identifying or categorizing an object/observation, is currently one of the most studied topics in computer vision and machine learning. We can break down the classification problem into four types: Binary, Multi-class, Multi-Lable, and Imbalanced Classification [33]. Binary classification is one of the straightforward classification problems. However, in the case of multi-class classification, the number of samples to train a model with high accuracy and the complexity of providing a prediction depends on the number of classes [44].

A quality dataset on Bangla sign language is really in need. However, until now, only a few alphabet-level sign language datasets are available, while word-level Bangla sign language is even more scarce. There are 38 hand signs representing 51 alphabets in Bangla sign language [41], and these 38 alphabets can create any word possible. On the other hand, a word-level Bangla sign language dataset MVBSL-W50 contains 50 classes of words [49]. A word-level classification model could be user-friendly but

may take a lot of computational resources while training and physical memory in real-life implementation.

1.2 Background Information

For socialising, communication is a crucial skill. Thomas Hopkins Gallaudet has worked to refine American Sign Language for more than three centuries. In Hartford, Connecticut 1817, Gallaudet University was founded as American Asylum, the first national school for the deaf community in the United States. American Sign Language is now one of the most commonly used languages in the American judicial system, and its use has spread across many countries [31].

However, in both West Bengal and Bangladesh, people try to follow their Sign Language, WBSL and BdSL. Calcutta Deaf and Dumb School Est. 1893 developed BdSL. Even though both WBSL and BdSL were developed as a single sign language, the Indo-Pakistan partition to the Independence of Bangladesh in 1971 caused them to diverge [17]. A utilises the BdSL49 dataset, which consists of 29,490 images of 49 individual Bangla alphabet signs in the Bangla sign language (BdSL). The dataset includes images of 14 adult individuals with diverse backgrounds and appearances. The dataset has been carefully prepared, and various noise elimination strategies have been employed [43].

Detection problems such as Sign language detection start with collecting data and creating a dataset. A lack of training data can produce a faulty and poor approximation.To tackle this problem, a dataset consisting of the largest image database for BdSL Alphabets and Numerals designed to reduce inter-class similarity while dealing with diverse image data that includes various backgrounds and skin tones, was introduced. Researchers suggest dynamic or continuous non-verbal communication. Continuous sign language is considered sentence-level American sign language. It is also suggested to perform recognition faster [34]. It should be kept in mind, sign language in different languages may provide different outcome for similar approach.

Testing with a simpler single and simple model, such as Inception V3 and RNN, causes a problem in detecting facial features. The accuracy can drop if data is not trained on certain skin tones [23]. Feeding multi-modal information to the system can overcome such problems. In order to complement the models, depth and RBG modalities are taken into account and merged, which resulted in the top score for the challenge of RGB and RGB-D of the SLR [39].

1.3 Research Objective

Our study aims to achieve better accuracy in recognizing Bangla Sign Language and translating it to human readable language, in this case, Bangla. Datasets that include images of different hand gestures for BdSL are fed to multiple CNN models. Then the proposed CNN models should help us identify BdSL with better accuracy by taking the average outcome of different CNN architectures.
To achieve our objective, we have to:

I. Use a dataset containing many images of different hand gestures of BdSL.

II. Utilizing three suitable CNN models.

III. Consider the average output score from different models to achieve better accuracy from all models.

We will use a dataset that contains images of different hand gestures that indicate different alphabets. There are multiple images for identical expressions signed by different native signers that will help our model to learn more efficiently. We will then process these data and feed them to our models. Following an ensemble approach, we will train multiple CNN models. Each CNN model will be architected with a suitable sequence of convolutional layers, max-pooling, and fully connected layers, and finally, average the output score from all the models. By taking the average output score from the weak classifiers, we will achieve improved predictive performance and attain higher accuracy rates for recognizing the sign gestures of BdSL.

1.4 Research Contribution

In this paper, we propose a novel procedure-based sign language recognition system. We trained light CNN models with three modalities to overcome environment-dependent obstacles such as lighting issues, skin color, and different hand structures. After training the models, we checked cross-checked validation results with our model individually with many state-of-the-art image classification models like Inception-V3, DenseNet, Xception, VGG19, and ResNet50. Then we tested multiple meta-learning algorithms for the ensemble model and crossed checked the testing data with all mentioned state-of-the-art classification models. The main contributions of this research are as follows:

- We propose a system where pre-processing technique takes precedence over the model's architecture. This paper provides extensive details on how we can pre-process samples so the classification model can provide a prediction with a more meaningful explanation.

- We propose a novel procedure-based system to tackle environment-dependent uncertainty variables like skin color, different lighting, noisy background, and hand structure. By reducing the uncertainty variables, classification models can provide a prediction with higher precision. This approach was inspired by how humans classify a hand sign, which considers modalities such as skin color, skeleton position, and the edges of the hand.

- We propose three lighter CNN classification models instead of a heavy neural network architecture for an efficient system. A lighter CNN model has fewer parameters, requires less computational power, and is faster and more efficient than their popular counterpart. Even in terms of training the models, it will require less resource utilization. Our proposed models performed better at generalizing unseen data as they are less prone to overfitting while training.
- We propose a particular type of ensemble meta-learning algorithm for the best combination of output where the meta-learning algorithm is integrated with the training average precision value of the models. We tested out five different modified meta-learning algorithms for predicting the best outcome. This approach also reduces any weak predictions by considering all the outputs from the models.

Chapter 2

Literature Review

2.1 Gesture Recognition

During predictions, a CNN architecture-based model can produce a high variance, and during training, overfitting can occur. An ensemble approach was suggested to reduce such inconvenience. The models were trained with background-separated images extracted using the binary thresholding technique. The models used for the ensemble models are a modified version of CNN-based classifiers such as VGG16, GooLENet, and AlexNet. The output accuracy of the models is then calculated and averaged. A meta-learner is built for the final gesture classification. Two publicly available databases, Dataset-i and Dataset-ii, and one custom dataset have been considered for the project. The results achieved were Dataset-i for 99.80%, Dataset-ii for 96.50%, and 99.76% on the self-constructed dataset [46].

2.2 Sign Language Recognition

A study shows an efficient technique for identifying Indian Sign Language(ISL) numbers, letters, and words used in everyday life using a Convolutional Neural Network, an architecture consisting of convolutional layers, ReLU layers, and max-pooling layers. A dataset of 35,000 samples from 100 static signs was created using a camera under various environmental conditions. The suggested architecture has been evaluated on around 50 deep-learning models using several optimizers. The grayscale picture dataset and the colored dataset achieved a training accuracy of 99.72% and 99.90%. In the performance comparison, SGD demonstrated superior results over Adam and RMSProp optimizers. The proposed system's outcomes have also been analyzed based on accuracy and other evaluation metrics. It was discovered that their model surpassed other existing models despite having fewer epochs [35].

A proposed method for the CSL recognition system is to segment the upper body part from the video, then send the segmented data to a pre-trained classification model to recognize the gesture. The procedure depends on the HSV transformation, so it removes the facial segment of the photo. Without the assistance of motion capture devices, the technique has produced extraordinary recognition accuracy of 99% on video on their self-build dataset, which contains 40 videos of CSL daily vocabulary [22].

An approach makes use of co-independent data streams to capture the various sign language modalities. These modalities share a complicated temporal structure and are represented by various information channels. To address this, the authors focus on synchronizing and capturing the dependencies between the sign language components. Handshapes are the critical components in sign interpretation, and putting them in the proper context is essential to understanding what a sign is trying to say. By incorporating an attention mechanism, the model effectively combines features of the hand with their relevant spatiotemporal context, leading to enhanced hand-sign classification. The authors emphasize the significance of identifying the essential sign language components associated with the dominant hand and face areas. Except for the image embedding layers, which were already trained on ImageNet, the network layers are initialized using Xavier. Regarding performance, the error rates on the dev and test sets are reported as 32.74% and 33.29%, respectively [42]. Songyao Jiang et al. propose the novel SAM-SLR-v2 (Skeleton Aware Multi-modal Framework with a Global Ensemble Model). The paper discusses the shortcomings of current SLR techniques, which frequently experience overfitting as a result of scarce and noisy data. The suggested framework combines multi-modal feature representations to increase SLR accuracy. A Sign Language Graph Convolution Network (SL-GCN) is first introduced to capture the dynamics of skeleton key points. In sign language, it represents hand gestures and body positions. The temporal relationships between important points are modeled with the aid of this network. This study used five distinct datasets, five of which included various languages. Three separate SLR datasets are used to evaluate the proposed SAM-SLR-v2 framework. The framework achieves state-of-the-art performance with significant improvements over existing approaches [40].

2.3 Bangla Sign Language Recognition

Older approach to Bangla sign language recognition employs a pattern-matching method based on PCA (Principal Component Analysis) for recognising 6-Bengali vowels and 10-Bengali numerals. This system acquires a picture using a CCD camera at 30 frames per second. The system has been taught to detect 16 hand signs, including 10 Bengali digits and 6 Bengali vowels. Bengali vowels and numbers are each given their own PCA. This program can recognise Bengali numerals from 0 to 9 written with one hand and Bengali signs with two hands. The highest precision rate for Bengali sign language is 98%, and for numbers, the rate is 87% [5]. The proposed system discusses the Neural Network Ensemble method, which does the training faster and recognizes the signs with 93% accuracy of generalization ability. For pre-processing, the system captions the image using a webcam, converts the image to a threshold image and uses the normalization technique for feature extraction. Then the matrix representation of the extracted image is sent for Negative correlation learning. After computing the training error and running iterations for the acceptable error, we stop training and run operational steps for BDSLR. NCL with 10 NNs with feature extraction comes to an accuracy of 93% [6].

Abedin et al., propose a new architecture called the "Concatenated BdSL Network" that combines a Convolutional Neural Network (CNN) based image network with a pose estimation network. The proposed approach aims to improve recognition accuracy by incorporating visual features and hand posture. The novel approach scored 91.51% on the test set, indicating promising results. The additional pose estimation network proved beneficial in dealing with the intricacies of BdSL symbols, as suggested by the experimental outcomes [36].
A comprehensive dataset for Bengali sign language was created and trained using CNN. Study demonstrates how convolutional neural networks may be utilized to identify various BdSL indications accurately. In the suggested model, testing accuracy for numbers was 100%, and testing accuracy for the Bangla sign language alphabet's characters was 99.83%. Combining characters and numbers achieved an accuracy of 99.80% [28]. The relative placements of the separate fingers vary according to the letter. As a result, fingertip position values are beneficial for identifying letters, and ANN is trained to use position values for this purpose. Using the Ravikiran approach, the Canny Edge Detection Algorithm finds the fingertip. The process initially identifies the edges, and then the exact position of each finger's tip is gathered. For categorizing the letters, ANN is taken into consideration due to its effectiveness in particular works. The ANN is trained to fix the iteration using the backpropagation training approach. According to observations, the accuracy of the training set for tip position-based BSL-FTP is comparable to that of the GSI-based technique. BSL-FTP, however, performs better than grayscale images on a general basis. BSL-FTP for 100 hidden neurons shows the highest test set classification accuracy of 98.99% [13]. Shahjalal Ahmed et al., examined their CNN model's evaluation using multiple datasets in the paper. The model classifies digit-based hand signs with 92% accuracy. The model consists of two convolution layers with 2x2 max pooling for each layer. The model was trained for 100 epochs with RMSProp optimizer, and CCE(Categorical Cross Entropy) was used as the cost function [27].

A paper on the framework for BSL recognition using a Support Vector Machine goes as follows, the framework works in 2 steps:

I. Segmentation of hand signs

II. Recognition of hand signs by extracting the features from the hand.

For the segmentation of hand signs, The research converted it to RGB to HSV and later HSV to Binary for Morphological operation. Moreover, for Sign recognition, the study segmented the axis of the regions of the images and extracted the features using a two-dimensional Gabor filter. After using the MATLAB tool to compute the success rate of the experiments, the total success rate came to 99.5% with 2389 correctly recognized images from 2400 images [19].
Another research shows the implementation of a Bangla Sign Language system that utilizes SIFT for feature extraction to enhance accuracy and CNN for classification. The research manually collected 200 images for 38 Bangla signs and 51 Bangla letters for training and testing purposes. The research team then implemented a skin masking technique to select the Region of Interest (ROI) containing only the

hand image. The feature descriptors were extracted using Scale-Invariant Feature Transform. The extracted features were treated as clustered descriptors using k-means clustering to create a histogram vocabulary for visualization. The data was inputted into the CNN as histograms, and the output was checked for accuracy, which was later compared with the results obtained without employing SIFT. The results obtained with SIFT operations demonstrated a significant improvement over the results obtained without using SIFT [25].

2.4 American Sign Language Recognition

The biggest challenge in ASL recognition is tackling the enormous ASL vocabulary that is ever-growing [2]. The system developed to recognize a large group of American Sign Language consists of a pair of task:

I. The stage for feature pulling or extraction.

II. The categorization phase.

A camera captures an image of the sign, eliminating the need for gloves. The feature extraction stage relies on the Hough transformation, which is lenient and unbiased of incomplete feature boundary information and comparatively insensitive to photo distortion. These extracted feature vectors were then used as input for the neural network. The proposed system demonstrated resistance to alterations in hand sign, placement, dimension, and coordinates. This is because the stage for feature extraction used techniques such as rotation, scale, and translation. The suggested system identified approximately 98.5% and 80% of the learning and testing sets [4]. The paper is based on an easy-to-use sign language recognition system. In order to tackle limitations due to the simplicity of the service, the authors suggest two types of models. It proposes the Inception model to bring out the spatial features of the video stream for SLR. It proposes an RNN model named LSTM in two methods to bring out the temporal information from the video. First by using the softmax's output and then by the pooling layer of CNN. For 150 signs, the accuracy for the softmax layer stood around 91%, and for the Pool layer at about 55%. The results of the model were based on a custom American Sign Language Dataset [23].

Another study offers an architecture for developing a ConvNet, which laid hold of depth and intensity data as distinct inputted data. The ConvNet was executed on the Torch platform and then assessed using an ASL benchmark dataset. The evaluation demonstrates that the developed convolutional network transcends previous research with 82% precision and 80% recall. Examination of the evaluation's confusion matrix reveals a pair of errors

I. The Symmetric errors: Where a pair of letters is misidentified as one another.

II. The Asymmetric errors: Where a of letters is misidentified as other but not vice versa [20].

Research suggests an effective feature extraction method utilizing a convolutional neural network (CNN). The experimental set-up was created from the beginning

to record a video frame from the camera. A suitable region is established as an ROI or Region of Interest to remove the undesirable portion of the video frame. Each individual has 3,120 (26x120) photos, and 120 photographs for each hand sign. 4096x2808 features will be available for training, and 6552x4096 features will be available for testing after feature extraction from the pictures using DCNN. The informational nature of each of these features assists in the classification of each human sign. Every alphabetic hand sign was categorised in the last phase of this suggested model using a non-linear Multiclass Support Vector System (MCSVM). The classification accuracy of 94.57% is noteworthy [24]. The HSV colour model plays a crucial role in specifying the boundary of skin from the background. A colour system named YCbCr is used to develop the functioning of skin colour clustering and construct a skin colour model of the hand. After comparing the ASL recognition rate of the suggested system with the existing ASL alphabet identification techniques, it is seen that this approach yields an accuracy of 93.05%, which is better than all the other existing techniques. Moreover, for ASL number recognition, the proposed technique has an accuracy of 95%, which is the highest compared to the pre-existing models in the past. It is seen that the average identification score of non-occluded hand gestures of ASL letters is 97.5%, and ASL numeric is 100% [26].

A model established on an iterative attention mechanism that focuses on ROI synchronously with increasing levels of resolution was studied. The ChicagoFSWild dataset is used for this research, which contains nearly 2400 finger sign orders, and four native signers provide them. Additionally, ChicagoFSWild+, a new dataset, was presented. This dataset was gathered using a fingerspelling gloss-video interface procured from VATIC. The ALexNet pre-trained on ImageNet is the foundation for the CNN model layers used in this case. Moreover, the last max pooling layer is dropped to achieve a large feature map. The recovered feature map is 13 x 13 when the input photos are 224 x 224 in size. A one-layer LSTM network with 512 hidden units is also used for the RNN. Next, Stochastic Gradient Descent (SGD) trains the model [32]. A depth map and black wristband can also significantly reduce the complexity of hand segmentation in hand sign recognition. A study was conducted on building an automatic fingerspelling recognition system using CNNs from depth maps. An accuracy of 78.39% for regular training and 85.49% for fine-tuning can be achieved via this mode [8].
A CNN is taught to recognize signs from static sign language pictures using a Leap-Motion Controller(LMC) to capture the coordinates of each hand joint for dynamic sign language collection. The data is later engineered to acquire relevant relative motion data and trained classical classification models to determine each LMC input's sign. The system will improve deaf and hard-to-hearing community communication. The final dynamic sign model can recognize one-handed signs with 88.9% accuracy and two-handed signs with 79% accuracy [37]. Combining HOG and LBP characteristics can extract the features from all the gestures. A multi-class SVM and a CNN were then trained on these extracted features. A combination of CNN and SVM models has proven to prevent overfitting, even if its validation accuracy is lesser than the HOGLBP-SVM model. The outcomes of the CNN and CNN-SVM models demonstrate that utilizing CNN as an independent feature extractor yields superior results compared to employing an end-to-end CNN architecture [29].

2.5 Word-level Dataset

This paper presents an extensive dataset for American Sign Language (ASL) at the word level, referred to as the WLASL dataset. The dataset contains a large number of common words that are frequently used daily. WSASL is also one of the, if not the most extensive, public American sign language datasets based on the number of samples present in each class and vocabulary size. The paper also evaluates the dataset by the performance of the deep-learning method, such as 2D human pose and holistic visual appearance. WLASL2000 contains 2,000 glosses with 21,081 videos and 119 signers. The paper proposes a temporal graph convolutional network (TGCN) that considers the pose trajectories' spatial and temporal dependencies together [34]. Another research paper discusses the importance of a robust sign language recognition system. It helped to overcome the communication barriers for individuals who struggle with verbal communication. The paper acknowledges different challenges in sign recognition, such as similar gesture patterns, lighting and clothing variations, and the need for a comprehensive Bangla sign language video dataset. To address these challenges, the researchers created a dataset called MVB-SLW50 that contains 50 isolated words across 13 categories. They also developed an attention-based Bi-GRU model that captures the temporal dynamics of pose information in sign language communication. The model simplifies the recognition process and achieves faster performance by concentrating on movement information and disregarding body appearance and environmental factors. The reported accuracy of the model is 85.64% [49].

Chapter 3

Work Plan

Ensemble stacking is perfect for enhanced classification, clustering, and regression problems [38]. We already stated how we use the classification method to translate sign language to human readable language.

In order to increase the precision of our final prediction, ensemble stacking is a technique that combines the output findings or predictions of numerous deep learning or machine learning models on the same dataset. The base model and the meta-model make up the ensemble architecture. As weak learners, the basis models serve as the framework for our meta-model. The meta-model discovers the best method for combining base model predictions. A meta-model can be both an AI prediction model and a predetermined algorithm.

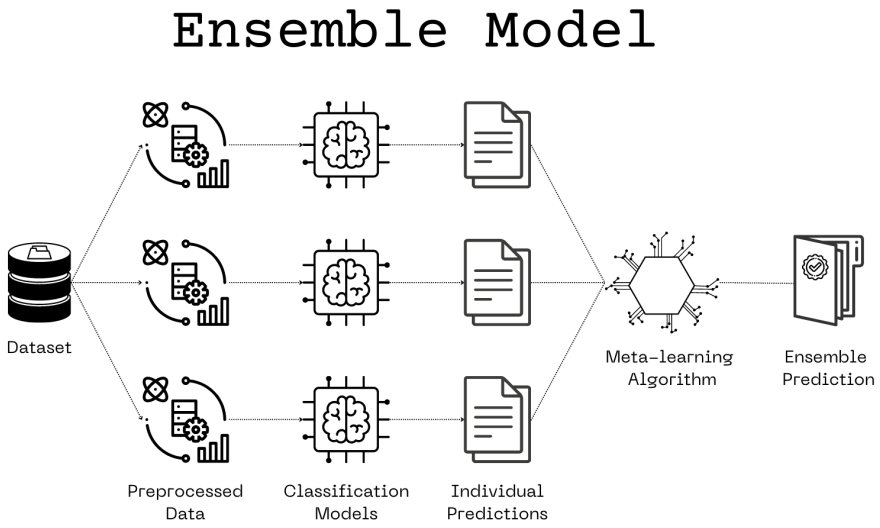

Figure 3.1: *Ensemble Model*

To understand the architecture of our work plan, we have divided our system into six different levels. Figure 3.1 is a visual representation of the ensemble stack model, and now we will go through each phase and describe the functionality.

Level 1: First, we choose a dataset suitable for all the classifiers. A good dataset for BdSL should contain more instances with consistent representation of signs.

Level 2: Before feeding our data into the system, we processed our data using pre-processing techniques such as Thresholding, GaussianBlur, Erosion, and Dilation.

Level 3 and Level 4: The pre-processed data is ready to be fed to the system. Each model was trained using the training set portion of the dataset. Each model here is a base learner.

Level 5 and Level 6: Using the meta-learning algorithm, we got a combined result from all the base learners. Furthermore, by level 6, the combined outputs from all the base learners should provide a better result.

3.1 Workflow

A dataset containing 11,061 training images over 38 different signs was taken for the research. Each sign is considered a class, and each class contains approximately 300 images. The folder includes images of hand signs that serve as the label. In this dataset, labels are represented as integers. Thus, it is necessary to go back and transfer the labels to the correct strings. After having a solid understanding of the dataset, the pre-processing stage starts to take place. In order to reduce the computational load on the computer, it was essential to downsize the photos from the dataset from 224x224 to 64x64. Then, augment each photo and split the new images with the old ones for the training and validation sets. For the testing set, an additional 1,520 photos were used for 38 classes. The dataset is now ready to put into the proposed model.

Now, creating a CNN model, use the training set and validation set to train the model. Various evaluation matrices were considered to determine whether the CNN model is adequate. Save the model in .h5 format after testing the model with the testing set and determining the best outcome. After saving it, we utilized our hand sign to check the model's functionality.

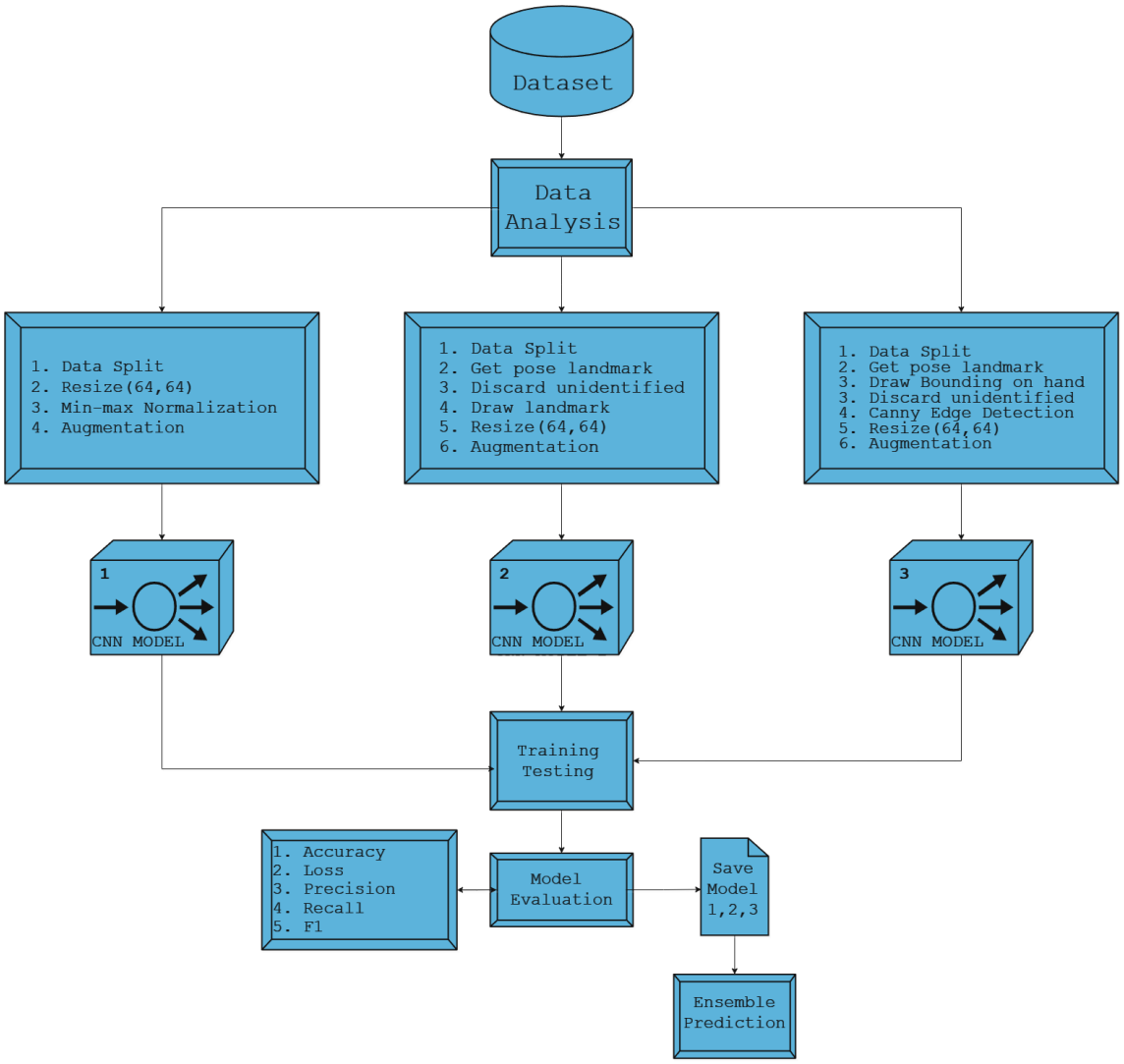

Figure 3.2: *Proposed Approach*

Chapter 4

Dataset

4.1 Data Configuration and Analysis

The dataset was referenced by Abdul Muntakim Rafi et al. in their paper on image-based Bengali Sign Language Recognition for Deaf and Dumb Communities [30]. The dataset contains 38 different hand signs from 278 different people. 38 different hand sign represents all Bangla alphabets, whereas in some cases, multiple alphabets are represented through the same hand sign. Most of the image contains a single-hand sign, and only one instance where both hands were used, which was expected since the dataset was on a single-hand sign bangla sign language. The image is mainly taken behind a white background, but a few hand signs are taken in front of a noisy background.

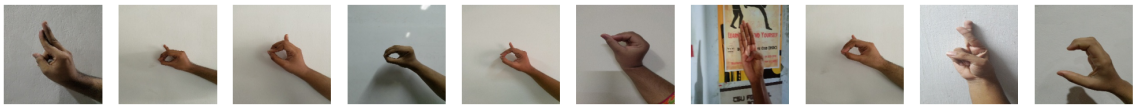

Figure 4.1: Random Images from the Training set

Figure 4.2: Random Images from the Testing set

4.2 Data Preprocessing

4.2.1 Model 1 Preprocessing Techniques

The pre-processing techniques limit the computational usage and make the model more accurate. A vast number of training data is required for a good model. Researchers suggest 1,000 instances per class is a good number to create a robust system, but only 320 instances per class are provided. So, we augmented each photo 6 times using rotation, width shift, height shift, shear and zoom. Then, we split our

11,061x7 images into training and validation sets. We used another 1,520 images of all 38 classes for our testing set. These 1,520 images were not augmented since they will evaluate our ensemble model. It is essential to ensure that no such augmentation occurs when we flip the photo horizontally, as it can change the meaning of the hand sign. After augmentation, we use the min-max normalisation [10] technique on our dataset. This will help us limit computational usage and provide faster predictions.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

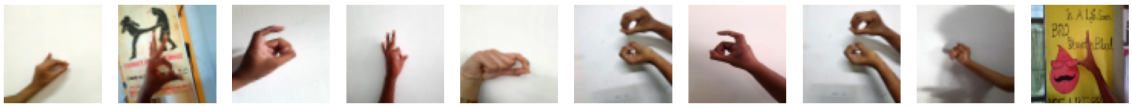

Figure 4.3: Random Images from the Preprocessed Training Set

Figure 4.4: Random Images from Preprocessed Validation Set

Figure 4.5: Random Images from Preprocessed Testing Set

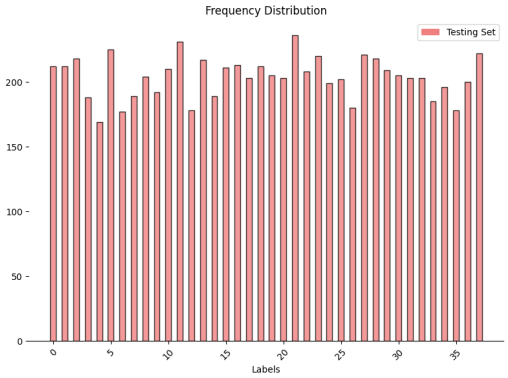

Figure 4.6: Validation Set

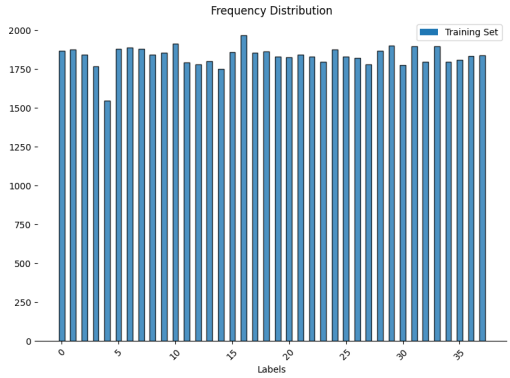

Figure 4.7: Training Set

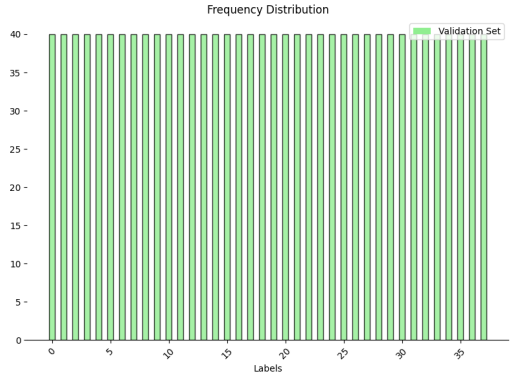

Figure 4.8: Testing Set

4.2.2 Model 2 Preprocessing Techniques

Figure 4.9: Pre-processing Technique Visual

Similarly, for model 2, augmentation will take place. But before augmentation, a few more steps are going to be required. For model 2, the focus is on the weakness of our model 1. Our model 1 can perform poorly when the background is very noisy, specially when background objects are larger than the arm or the hand in the photo. We also aim to use different modalities to get better predictions on the ensemble test. In order to achieve the goals, using MediaPipe's hand landmarks detection fits perfectly [48]. The workflow of the pre-processing technique for model 2 is simple. First, take the image and resize the image to (128,128). Use the hand landmark detection algorithm while setting minimum detection confidence to

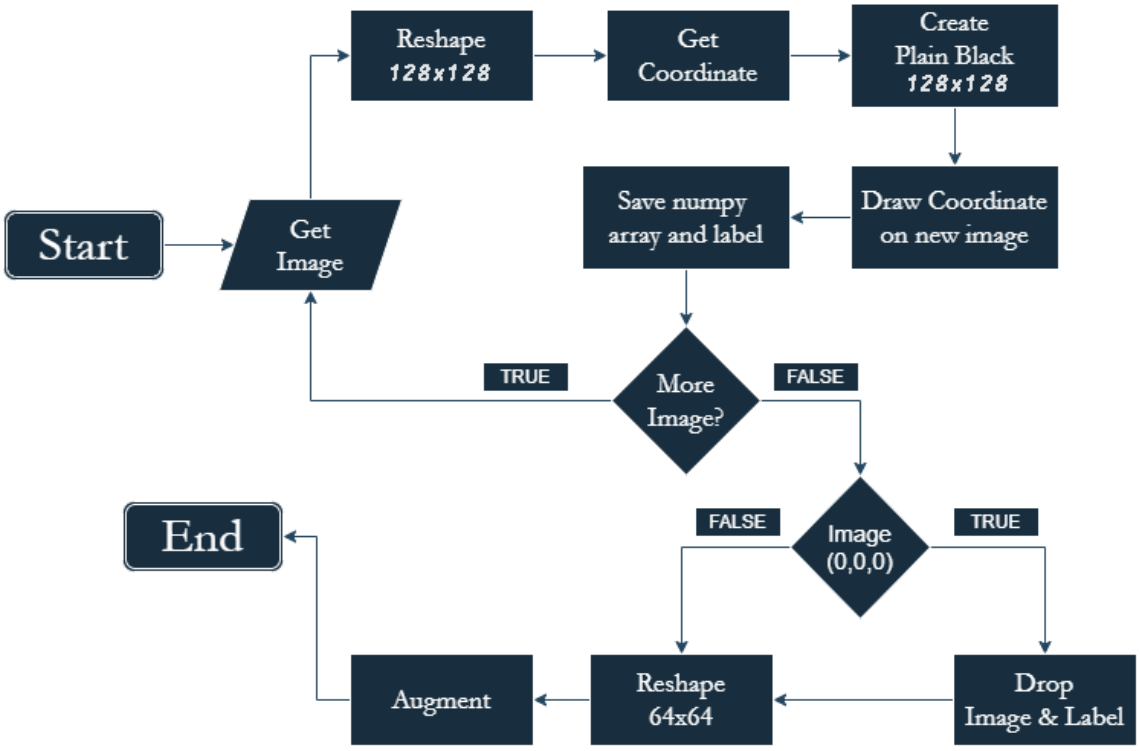

Figure 4.10: Flowchart Preprocessing Model 2

0. After fetching the coordinates for the hand and finger landmarks, draw the crossroads on the 128x128 image where every pixel value is set to (0,0,0). Then, resize the image to a 64x64 dimension. Sometimes, the MediaPipe cannot draw the hand pose, resulting in a completely black image. For such cases, discard the new entry. This same processing is required for all the images (11,061).
After creating the new pre-processed dataset, augment the images like the pre-processing technique for model 1.

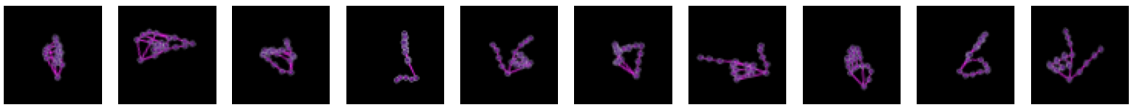

Figure 4.11: Random Images from the Pre-processed Training Set Model 2

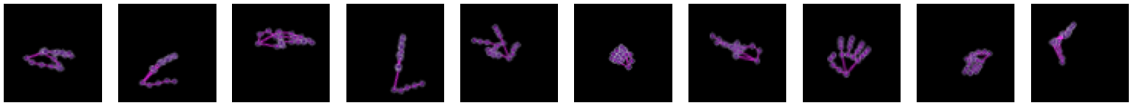

Figure 4.12: Random Images from the Pre-processed Validation Set Model 2

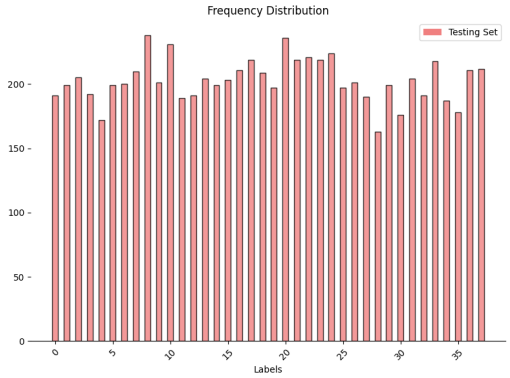

Figure 4.13: Validation Set

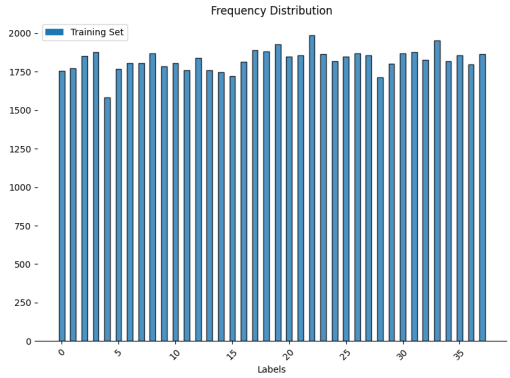

Figure 4.14: Training Set

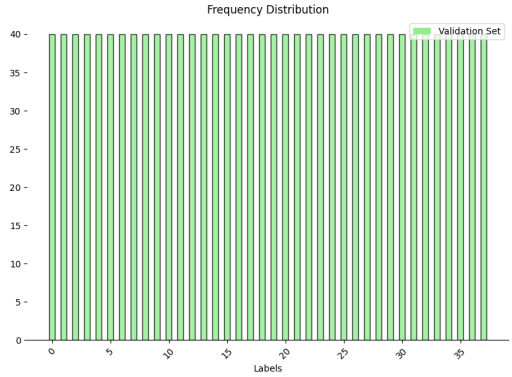

Figure 4.15: Testing Set

4.2.3 Model 3 Preprocessing Techniques

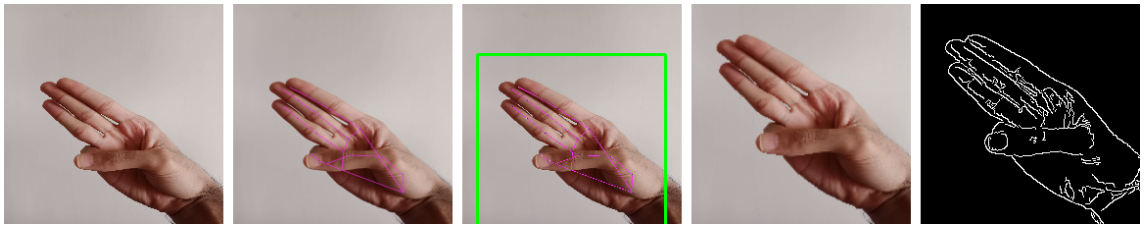

Figure 4.16: Pre-processing Technique Visual

Model 3 requires some extensive pre-processing techniques. The target for Model 3 was overcoming noisy backgrounds and removing complete dependency on MediaPipe. A problem with model 2 was that it could sometimes not find the hand and ended up forwarding a complete black image for prediction. To overcome the problem, the MediaPipe hand pose estimation technique was used to find the maximum and minimum x and y coordinates.
The total pre-processing flow is simple, but some small tricks are used to enhance the feeding data. At first, take a single input image with the label, then reshape the image with a dimension of (224,224). Apply MediaPipe hand landmark detection [48]. Filter out x-max, y-max, x-min, and y-min. Then crop the image at these four coordinates (x-min, y-min), (x-min, y-max), (x-max, y-min), and (x-max, y-max). If hand landmarks are not found, continue without cropping and reshape the image at (64,64). The cropping helps remove most of the background and bring the hand

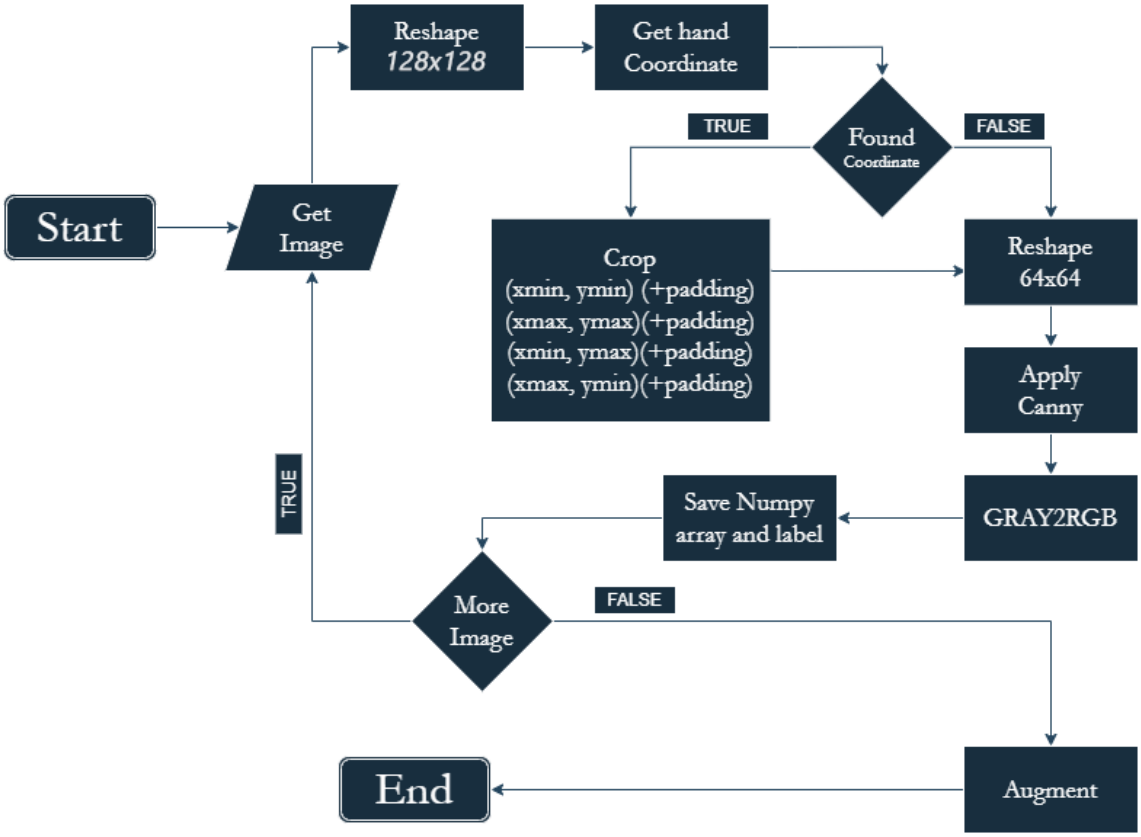

Figure 4.17: Flowchart Preprocessing Model 3

to the center. Apply the Canny edge detection algorithm [1] [51] . It not only serves as the edge detection technique but also drastically reduces the amount of data to be processed. Using canny changes the image to grayscale, so apply GRAY2RGB, then save the image and the label.

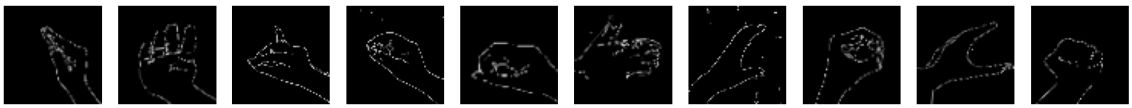

Figure 4.18: Random Images from the Pre-processed Training Set Model 3

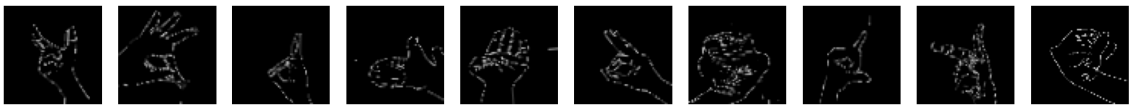

Figure 4.19: Random Images from the Pre-processed Validation Set Model 3

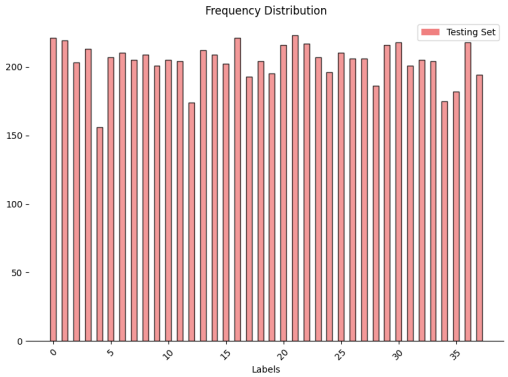

Figure 4.20: Validation Set

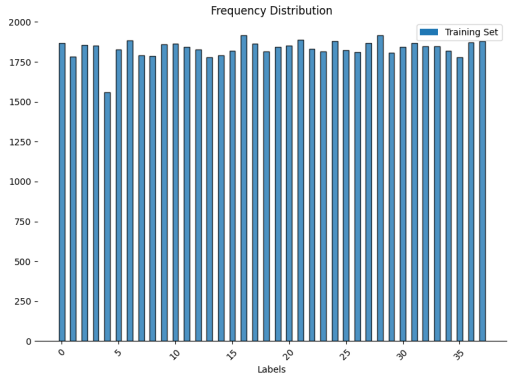

Figure 4.21: Training Set

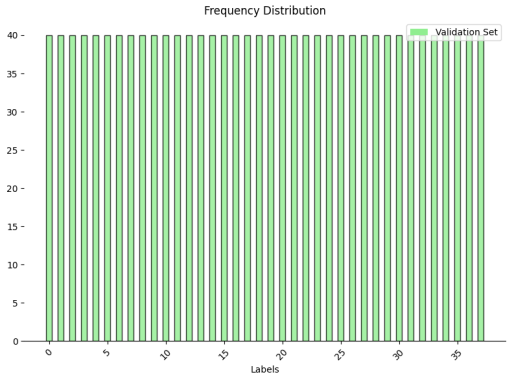

Figure 4.22: Testing Set

Chapter 5

Methodology

In this section, we described what we have proposed for Bangla Sign Language Recognition. The proposed model is based on the convolutional neural network architecture. Models built on CNN architectures are good at performing image processing challenges, such as image classification challenges. The convolutional neural network is designed to capture and exploit the hierarchical spatial structure present in images.

5.1 Convolutional Neural Network

The CNN is built on the same principle as the human brain. The brain's neuron and the convolutional neuron in CNN work similarly by sharing and communicating with local neurons. The CNN architecture usually consists of three layers: convolutional, pooling, and fully connected. The convolutional layer defines the resulting output of neurons connected to the input's specified region by calculating the scalar product between their weights and the region connected to the input volume [9].

Each layer of CNN works to detect different features from the input image. In order to complete our task of sign language recognition, layers have to learn the pattern of the fingers and the hand. A kerne creates an output that slowly gets better after each layer. The final layer, or the fully connected layer, provides the final output, which is the prediction of the hand sign for our case.

5.2 Proposed CNN Models for Sign Language Detection

In order to address the task of sign language recognition, we propose three distinct CNN models, each with its own characteristics and trade-offs. These models have been designed to cater to different requirements in terms of prediction speed and accuracy, allowing flexibility for various real-world applications.

5.2.1 Model-1

We propose a lightweight CNN model that predicts a hand sign fast. In terms of sign language recognition, the model needs to keep up with the signer if used in

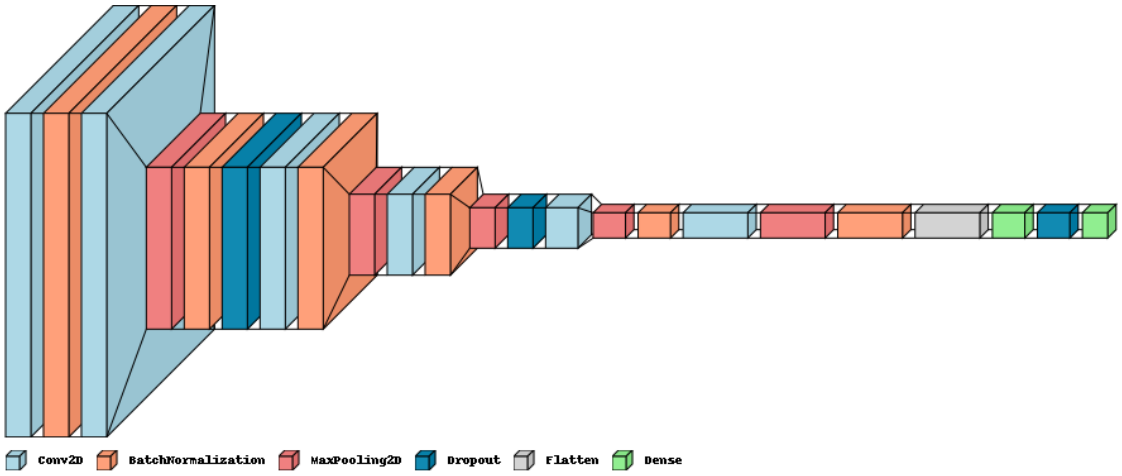

Figure 5.1: Illustration of the Proposed CNN Architecture for Model 1

real-time. So in order to make the prediction time faster, we have to rely on the dataset more than the model. If the model is too complex, predicting will take more time. A complex model provides output with better accuracy, while a simple model may predict faster. So it is essential to find a suitable middle ground. A model relies on the dataset too, so we went for a complex dataset with a simple model, so it learns the required features and predicts faster.

Our proposed model is built on nine different phases. Phase 1 consists of the Conv2D layer with a filter size of 32, a kernel of (3,3), ReLu activation and a batch normalization layer. It starts with extracting 32 feature maps using the kernel size. BatchNormalization normalizes the activations of the previous convolutional layer, improving the stability and convergence speed of the neural network. Phase 2 consists of the Conv2D layer with a filter size of 48, a kernel of (3,3), strides (1,1), ReLu activation, and a pool size of (2,2) for a MaxPooling2D layer with and strides (2,2). The configuration for kernel size, Conv2D strides and activations remains the same for all future phases. Phase 3 comes with five different layers, starting with the BatchNormalization layer, then the Dropout layer, Conv2D with the filter of 64, another BatchNormalization layer and finally, MaxPooling2D.

Phases 4, 5 and 6 all consist of Conv2D, MaxPooling2D and BatchNormalization layer, but for phases 4 and 5, the Conv2D has 128 and 256 filters while the remaining comprises 512 filters. Phase 7 is the Flatten layer, so the 3D feature map is converted into a 1D vector of length 512. Phase 8 is the first Dense layer, a fully-connected layer with 256 neurons and, again, a ReLu activation. Phase 9 has another Dense layer with 38 neurons corresponding to each of the classes of our hand sign but instead of using ReLu activation, it is required to use softmax.

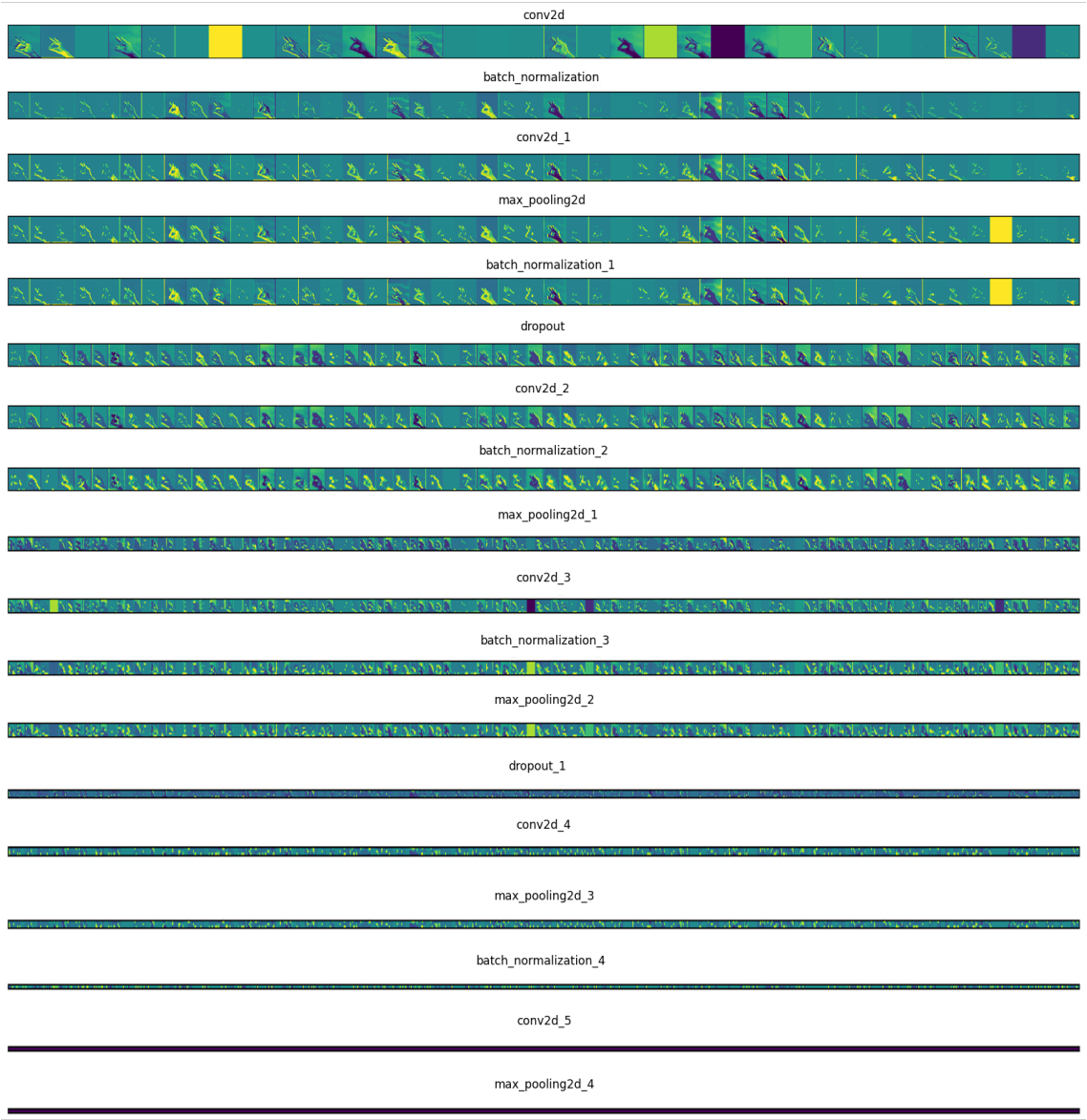

Figure 5.2: Feature Map on each layer of our proposed model

5.2.2 Model-2

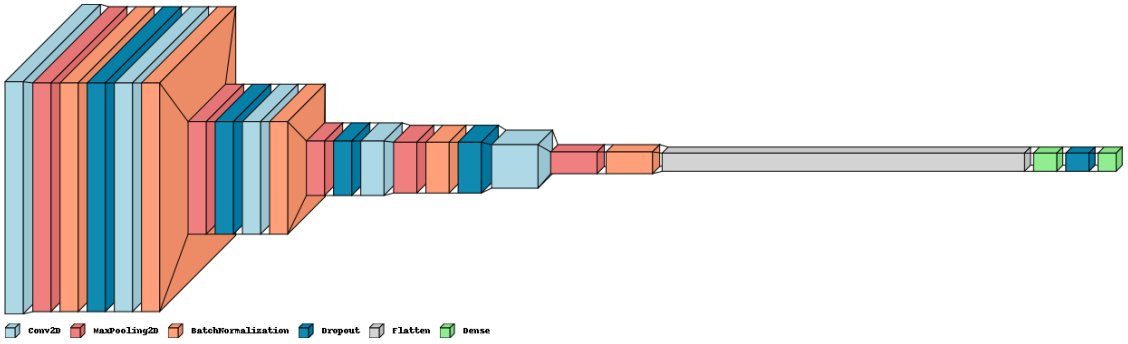

Figure 5.3: Illustration of the Proposed CNN Architecture for Model 2

The model starts with a Conv2D layer with 32 filters and a kernel size of (3, 3), using the ReLU activation function and employing padding to maintain the same output size. A MaxPooling2D layer follows the Conv2D layer, with a (2, 2) pool size and strides of 1, so the pooling window will move 1 step at the x-axis and y-axis. The subsequent layers continue this pattern. Again, a Conv2D layer is added with 64 filters and a 3x3 kernel, followed by BatchNormalization to normalize the activations and improve training stability. Then, a MaxPooling2D layer decreases the size of the feature maps' spatial dimensions. The process is repeated with Conv2D layers having 128 filters, 256 filters, and 512 filters. Each layer is followed by BatchNormalization and Dropout layers to regularize and further control overfitting. After the convolutional layers, a flattened layer is applied to convert the 3D feature maps into a 1D vector, preparing the data for the dense layers. The model then introduces two dense layers. The first dense layer consists of 256 neurons with a ReLU activation function. It is continued with a Dropout layer to regularize the model further. The last dense layer has units equal to the number of classes (num_classes) required to classify, employing a softmax activation function to produce the final probability distribution over the classes. During training, two callbacks are utilized. The ReduceLROnPlateau callback reduces the learning rate if the monitored metric (in this case, loss) stops improving, with a minimum learning rate of 0.00000001 and patience of 15 epochs. The EarlyStopping callback halts training if the monitored loss does not improve for 10 consecutive epochs. The model is constructed by employing the Adam optimizer with its default learning rate, utilizing categorical cross_entropy as the loss function for multi-class classification, and measuring its performance using accuracy as the evaluation metric.

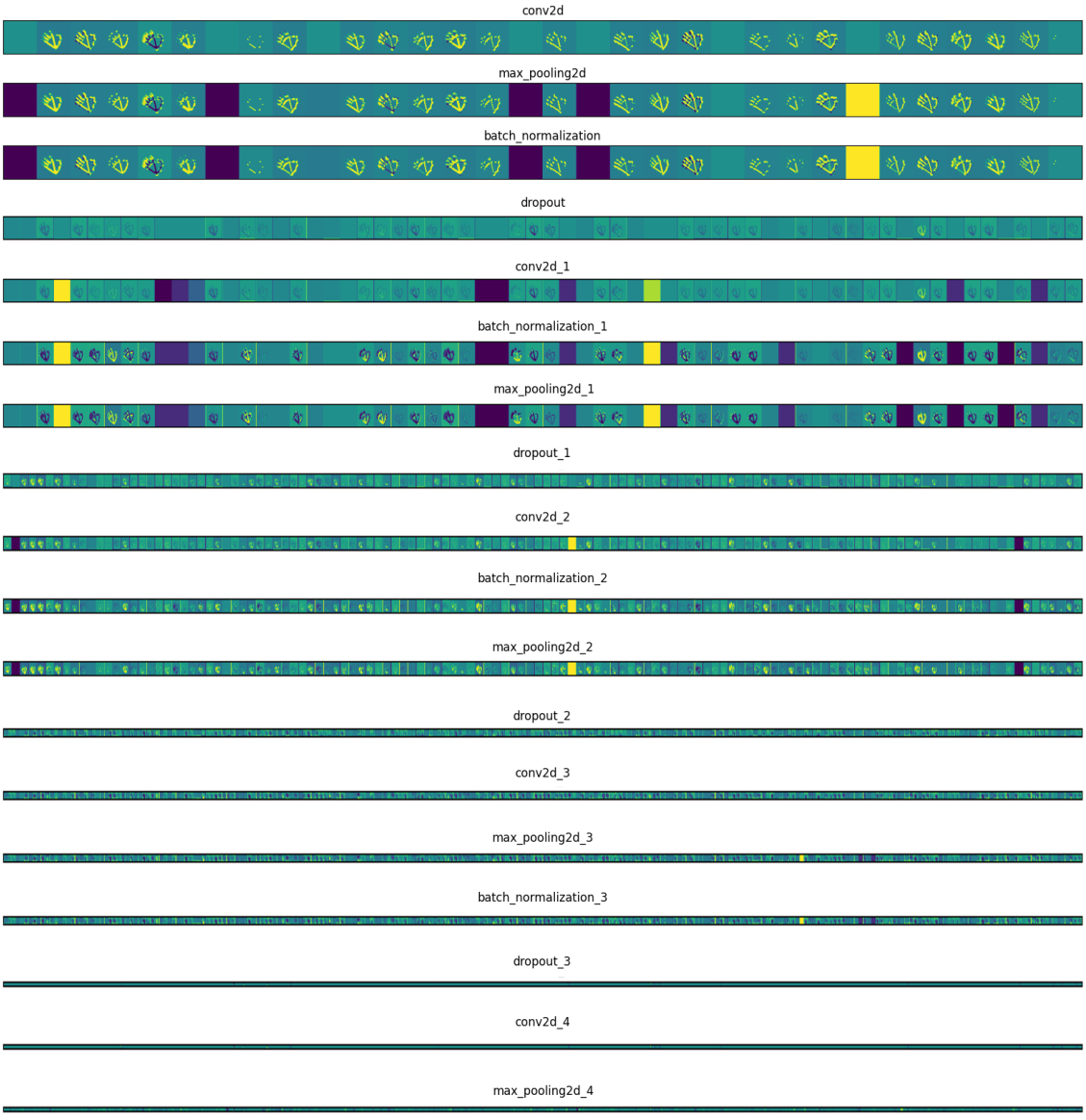

Figure 5.4: Feature Map on each layer of our proposed model 2

5.2.3 Model-3

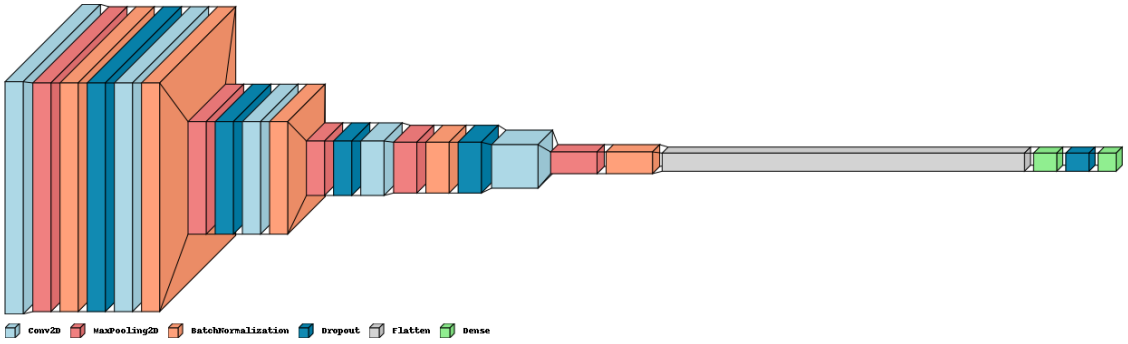

Figure 5.5: Illustration of the Proposed CNN Architecture for Model 2

In this model, convolutional layers begin with a Conv2D layer with 32 filters, a kernel size of (3,3) with ReLU activation, and a stride of 1. The padding is the same for all the layers. This is followed by a MaxPooling2D layer with a pool size of (2,2) and strides of 1. Batch normalization is applied to normalize the activations, and dropout with a rate of 0.1 is used for regularization. The sequence then continues with Conv2D layers with 64, 128, 256, and 512 filters following the same parameters. They are also followed by MaxPool and BatchNormalization layers along with dropout layers to regularize and further control overfitting. Finally, a Flatten layer is added to convert the 3D feature maps into a 1D vector. The dense layers begin with a Dense layer of 256 units and ReLU activation. Dropout with a rate of 0.5 is applied for regularization. Then, the model includes a Dense layer with units equal to the number of classes (num_classes) in the classification task. The activation function used is softmax to obtain class probabilities. During training, two callbacks are utilized. The ReduceLROnPlateau callback reduces the learning rate if the monitored metric (loss) stops improving, with a minimum learning rate of 0.00000001 and patience of 15 epochs. The EarlyStopping callback halts training if the monitored loss does not improve for 10 consecutive epochs. The model is compiled with the Adam optimizer using a default learning rate. The loss function is categorical_crossentropy, and accuracy is used as the evaluation metric.

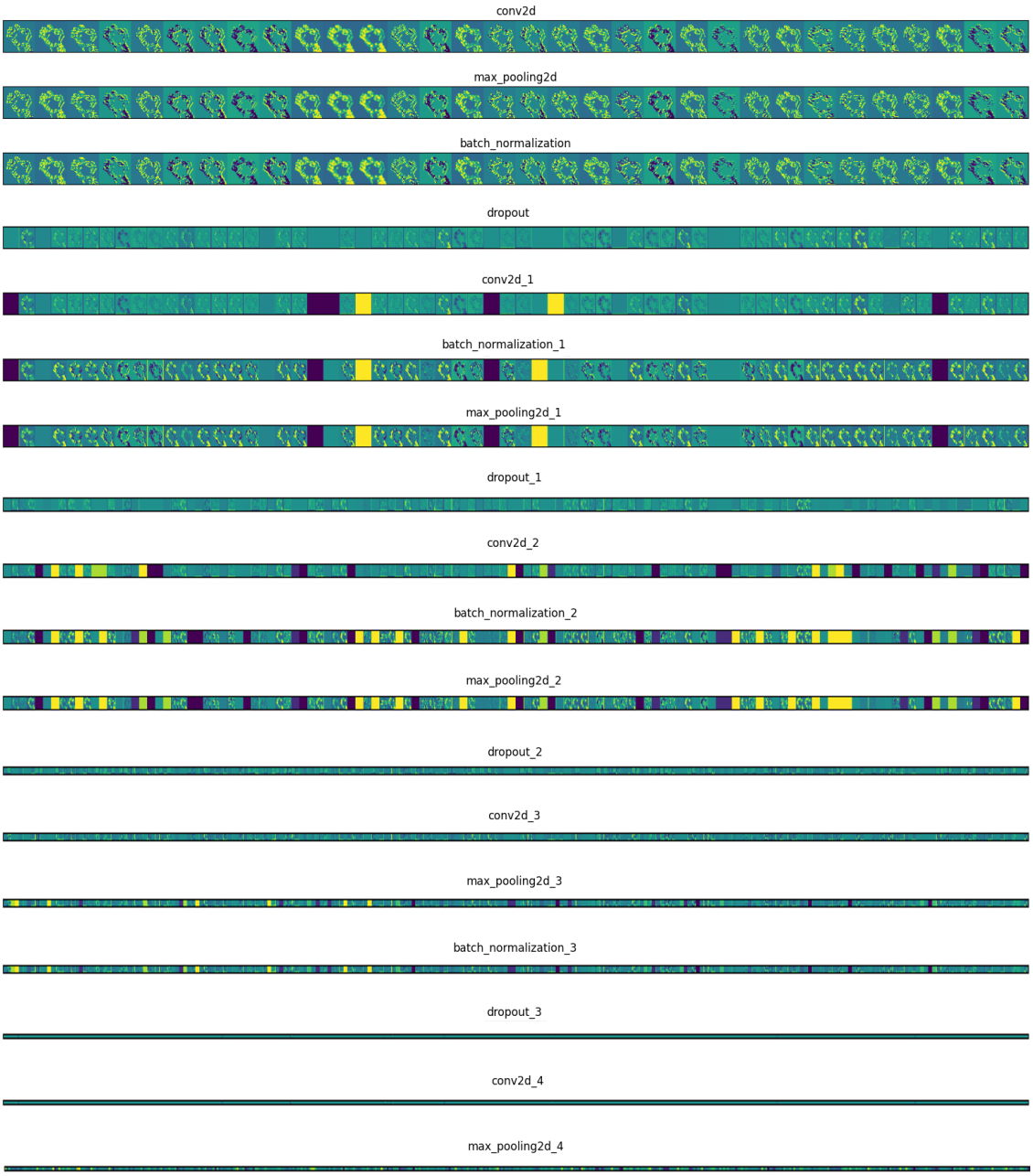

Figure 5.6: Feature Map on each layer of our proposed model 2

Chapter 6

Pre-trained CNN Models

6.1 VGG19

The architecture of VGG19 consists of 19 layers, including 16 convolutional layers and three fully connected layers. The convolutional layers use small 3x3 filters with a stride of 1, stacked together multiple times, making the network deeper. This stacking of convolutional layers helps the model learn more complex features at different levels of abstraction. Between the convolutional layers, VGG19 utilises max-pooling layers with a 2x2 filter and a stride of 2. A VGG19 model was suggested in another paper based on BdSL detection, which had achieved a testing accuracy of 89.6% [30]

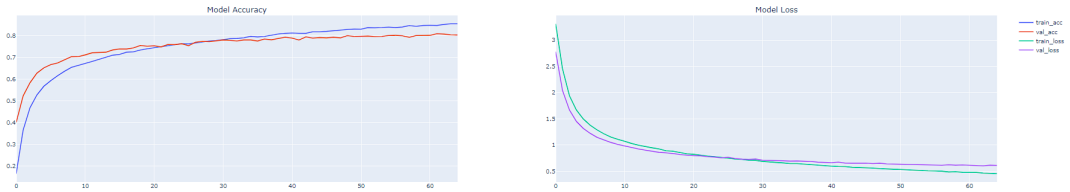

Figure 6.1: VGG19 Accuracy and Loss

6.2 Inception V3

The main idea behind Inception v3 is using "Inception modules," which are computational units that allow the network to capture information at multiple scales and resolutions. These modules incorporate different sizes of convolutions (1x1, 3x3, and 5x5) and pooling operations, enabling the network to extract features at various levels of abstraction. This multi-scale approach helps the model learn both low-level details and high-level concepts simultaneously. One notable feature of Inception v3 is the introduction of "factorised 7x7 convolutions." Instead of directly applying a 7x7 convolutional layer, the network factorises it into two consecutive 3x3 convolutions. This factorisation reduces the number of parameters and allows for better utilisation of computational resources. In Inception version 3, a significant focus is placed on optimising the computing resources required to execute the program.

This is achieved by implementing several modifications to the previous Inception architectures to minimise resource consumption [11].

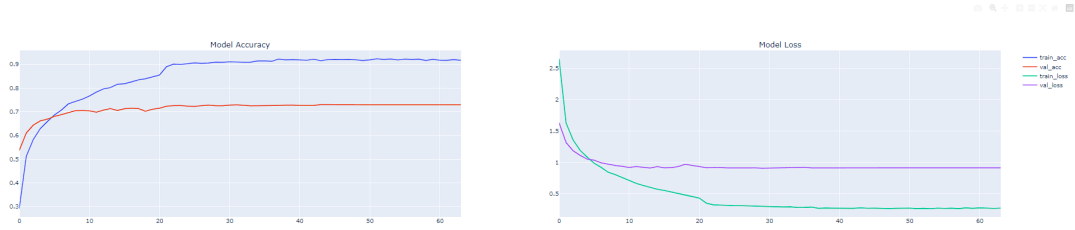

Figure 6.2: Inception V3 Accuracy and Loss

6.3 DenseNet

The key idea behind DenseNet is to connect all the layers in a feed-forward fashion [16]. The architecture of DenseNet is formed of many dense blocks, where each thick block contains a series of densely connected layers. Each layer receives inputs directly from all parent layers inside a dense block, creating a dense connectivity pattern. This dense connectivity makes feature reuse possible, as earlier layers have access to the gradients and features from subsequent layers, enhancing gradient flow and promoting information propagation throughout the network. To ensure efficient information flow and maintain a manageable number of parameters, DenseNet incorporates transition layers in the linking dense blocks. The transition layers are formed of a batch normalization layer, continued by a 1x1 convolutional layer, and a downsampling operation. These transition layers decrease the size of the feature maps and manage the number of feature maps, leading to a further reduction in computational demands.

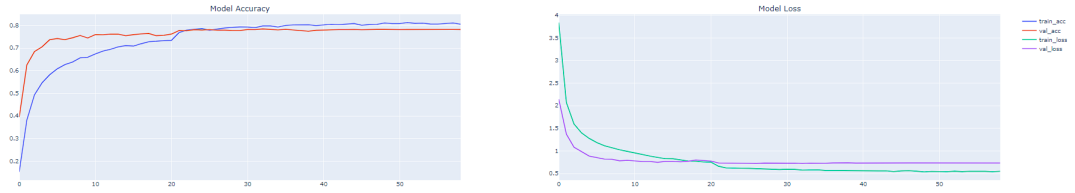

Figure 6.3: DenseNet Accuracy and Loss

6.4 Xception

A convolutional neural network architecture that exclusively utilizes depthwise separable convolution layers. The hypothesis is that convolutional neural networks' relationships between channels and spatial features can be separated completely. This hypothesis builds upon the idea behind the Inception architecture and takes it to an extreme level. Hence, we refer to our proposed architecture as Xception, which stands for "Extreme Inception" [14]. The main feature extraction component of the

model consists of 36 convolutional layers. These convolutional layers are organized into 14 modules, where each module is designed with linear residual connections, except for the initial and final modules.

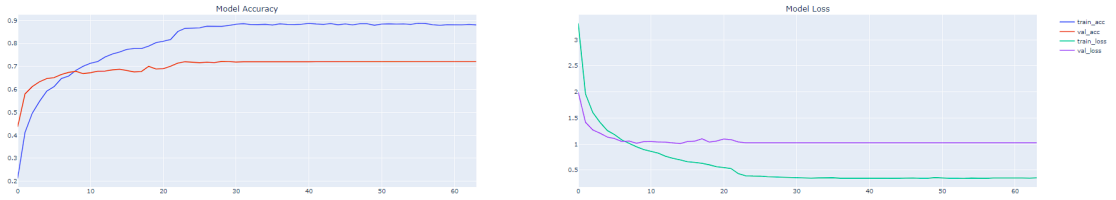

Figure 6.4: Xception Accuracy and Loss

6.5 ResNet50

ResNet50 is a 50-layer deep convolutional neural network with an input size of 224 by 224. We can load a pre-trained version of the neural network trained on more than a million images from the ImageNet database. It can classify images into 1000 object categories. The structure of ResNet is similar to VGG with residual blocks. It has a 7x7 conv layer and only one pooling layer with two-sized strides followed by numerous conv layers. Finally, an average pooling layer is followed by a fully connected layer with 1000 nodes using the softmax function. This CNN also uses 1x1 conv layers known as "bottleneck." Bottleneck residual blocks reduce the number of parameters and matrix multiplication, resulting in much faster training of individual layers [15].

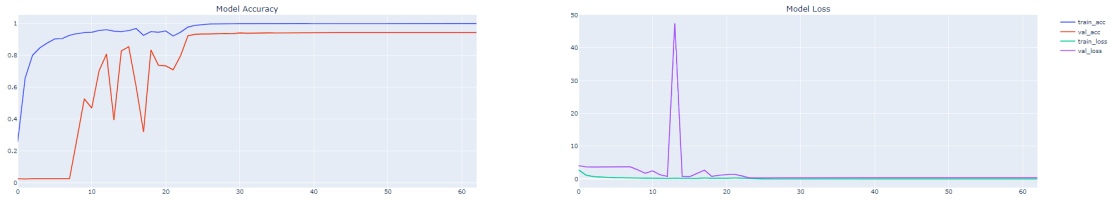

Figure 6.5: ResNet50 Accuracy and Loss

Chapter 7

Experiment Results

7.1 Results

A study comparing the proposed models with the five other pre-trained models was conducted, which included state-of-the-art models such as VGG19, Inception V3, DenseNet, Xception, and ResNet50. The proposed models achieved high assessment metrics values compared to the pre-trained models. Only ResNet50 could beat model 2, but it was expected as model 2 works on a different modality than Resnet50, adding that ResNet50 is a significantly larger architecture compared to model 2, and it requires samples with higher dimensions. This experiment incorporates evaluation metrics such as accuracy, precision, F1 score, and loss to evaluate the models' performance.

The accuracy of a model measures how often the model correctly predicts the expected outcome. It is the fraction of accurate predictions made by the model relative to the total number of predictions.

Precision is used to determine whether a percentage of the identifications made by the mode is accurate. To assess precision, we take the sum of all predicted positive outcomes known as True Positives (TP) and divide it by the sum of the total number of predicted positives (TP + FP) [3]. Hence, we use the following formula:

$$Precision = \frac{TP}{TP + FP}$$

The Recall metric measures how accurately a model can identify positive examples from a dataset. It is calculated by dividing the number of true positives (TP) by the sum of true positives and false negatives (FN). The formula for the recall is:

$$Recall = \frac{TP}{TP + FN}$$

Finally, the F1 score is used to determine the overall performance of the models. It is calculated by taking the harmonic mean of the precision and recall scores. We use the following formula to select it:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Table 7.1: Evaluation metrics for Training Set

Deep Learning Architecture	**Accuracy**	**Epochs**	**Recall**	**Precision**	**Loss**	**F1 Score**
VGG19	0.8561	65	0.9022	0.9041	0.4543	0.9018
Inception V3	0.9170	65	0.9876	0.9876	0.2703	0.9876
DenseNet	0.8052	65	0.9728	0.9729	0.5509	0.9727
ResNet50	0.9987	65	0.9999	0.9999	0.070	0.9999
Xception	0.8808	65	0.9881	0.9882	0.3529	0.9881
Model 1	0.9977	66	0.9999	0.9999	0.0075	0.9999
Model 2	0.9811	80	0.9995	0.9995	0.0753	0.9995
Model 3	0.9930	80	0.9994	0.9994	0.0201	0.9994

Table 7.2: Evaluation metrics for Validation Set

Deep Learning Architecture	**Accuracy**	**Epochs**	**Recall**	**Precision**	**Loss**	**F1 Score**
VGG19	0.8045	65	0.8045	0.8083	0.6119	0.8037
Inception V3	0.7297	65	0.7300	0.7307	0.9105	0.7293
DenseNet	0.7816	65	0.7816	0.7846	0.7342	0.7817
ResNet50	0.9416	65	0.9416	0.9421	0.3763	0.9417
Xception	0.7213	65	0.7213	0.7245	1.0248	0.7219
Model 1	0.9765	66	0.9766	0.9766	0.1005	0.9765
Model 2	0.9330	80	0.9330	0.9334	0.3309	0.9329
Model 3	0.9666	80	0.9666	0.9667	0.1591	0.9666

A comparative analysis with existing baseline or benchmark pre-trained models in the field was conducted. The evaluation metrics considered for this analysis included accuracy, precision, recall, and F1-score. The proposed models demonstrated a significant improvement in all the evaluated metrics, outperforming the existing pre-trained models. Specifically, proposed model1 achieved an accuracy of 97.65% on the testing set, surpassing the highest reported accuracy of the previous models. This substantial performance enhancement highlights the proposed model's effectiveness in addressing the problem.

7.2 Confusion Matrix

The confusion matrix of the validation set of our proposed model displays the predictions on the horizontal axis and the correct responses on the vertical axis. Confusion matrix is a process that determines the performance of the model by cross-checking the predicted value with the true value [45]. Figure 7.1, 7.2, and 7.3 contain the confusion matrix for the samples from the validation set for each model with a heat-map.

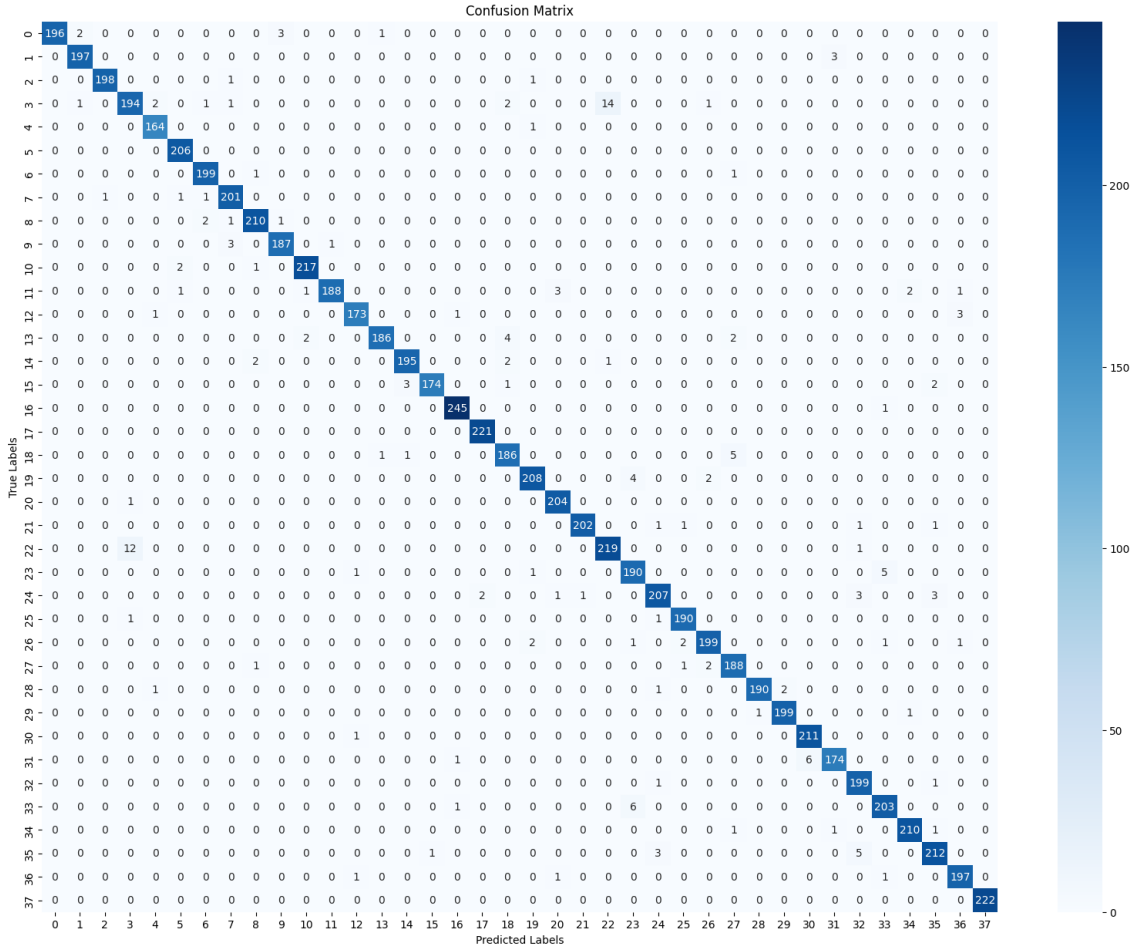

Figure 7.1: Confusion Matrix on Validation Data Model 1

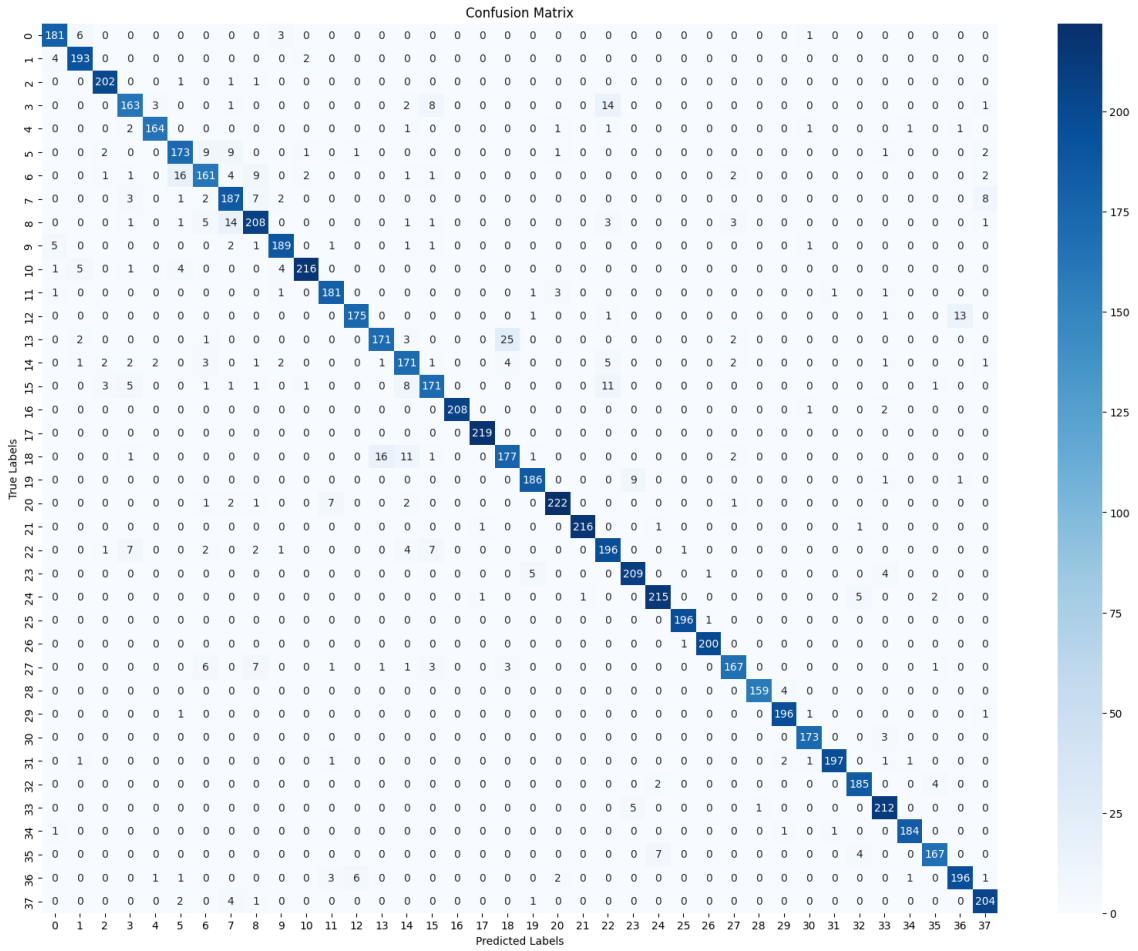

Figure 7.2: Confusion Matrix on Validation Data Model 2

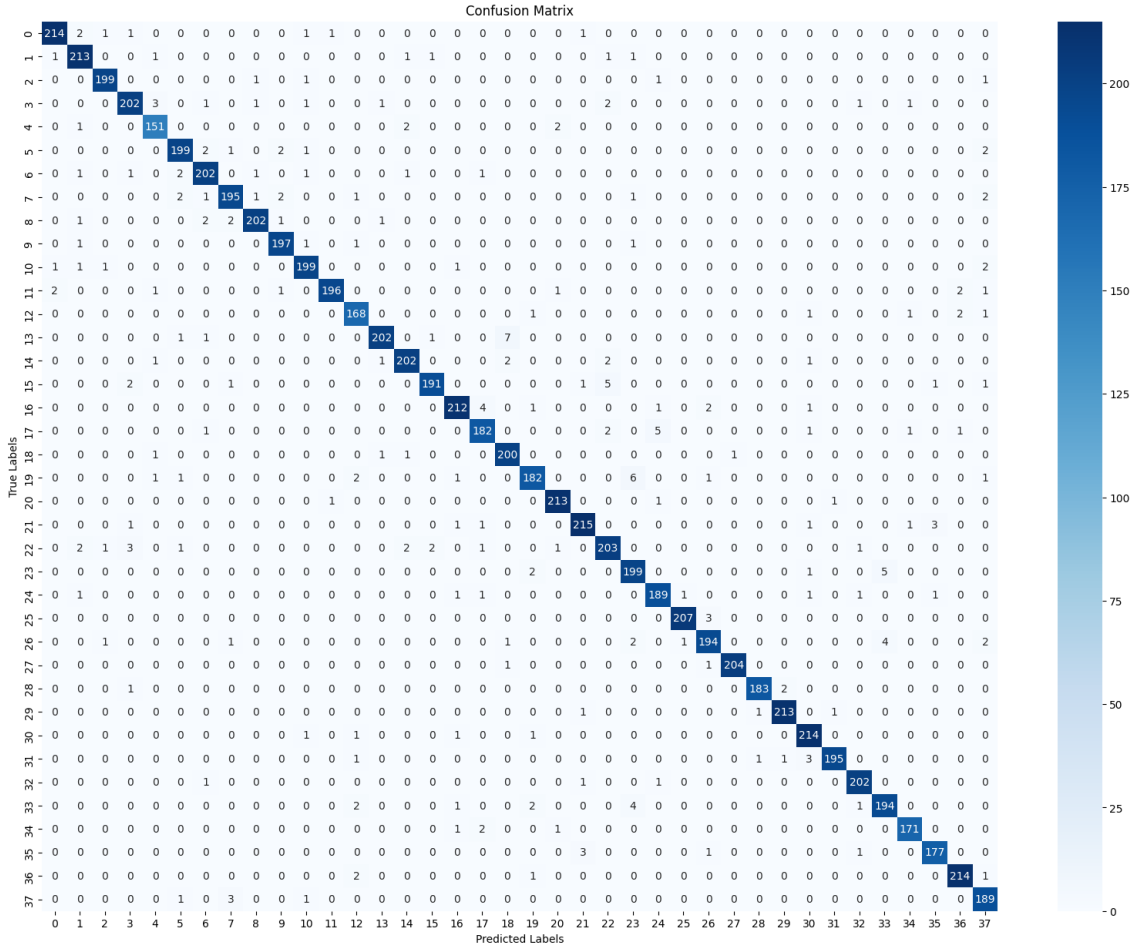

Figure 7.3: Confusion Matrix on Validation Data Model 3

Some hand signs are very difficult to generalise, specially when the hand signs are similar with just a few minor changes. Labels 3 and 22 are identical except for the angle of the index finger where most of the false positive occurred. The problem was tackled by use of model 3.

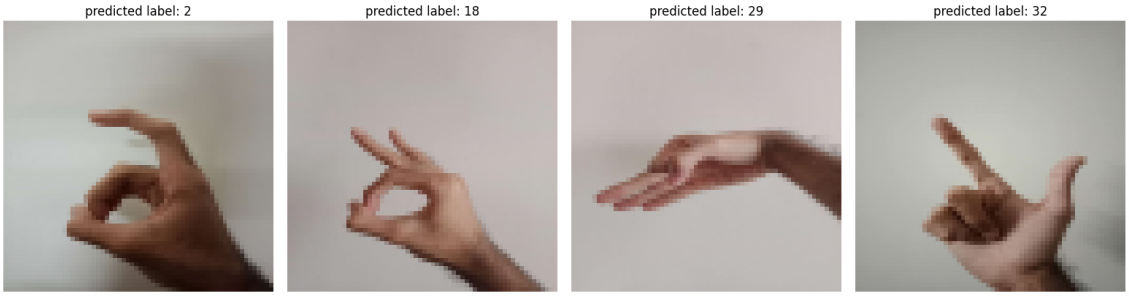

Figure 7.4: Predicting hand sign taken from the phone camera

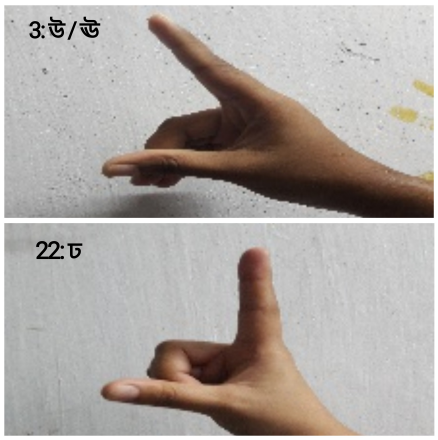

Figure 7.5: False Prediction between labels 3 and 22

In other cases, our model can provide a prediction fast and accurately. We took multiple photos of our hands with our cellphone camera and tried to predict the hand sign using our models, and it was able to provide the correct prediction.

7.3 Classification Report

The classification model for all three models is provided below. This report will help in figuring out the weakness of the models. It suggests that out of all the models. Model 2 had the lowest precision among all the models and the lowest individual precision for a single class standing at 0.83 for classes 7 and 14. However, in some cases, model 2 had the best precision, beating both model 1 and model 3.
The graphs of accuracy and loss of the proposed model are as follows:

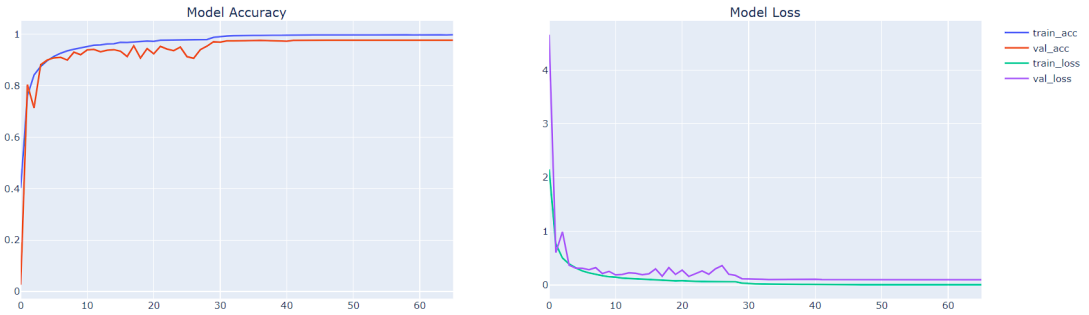

Figure 7.6: Proposed Model 1 Accuracy and Loss Graph

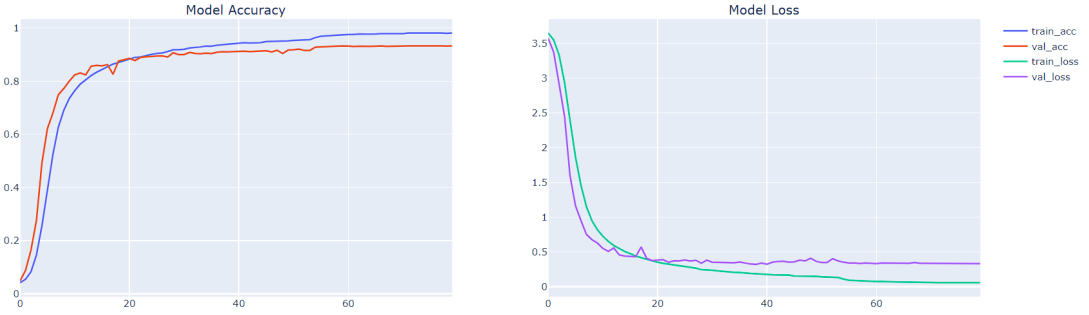

Figure 7.7: Proposed Model 2 Accuracy and Loss Graph

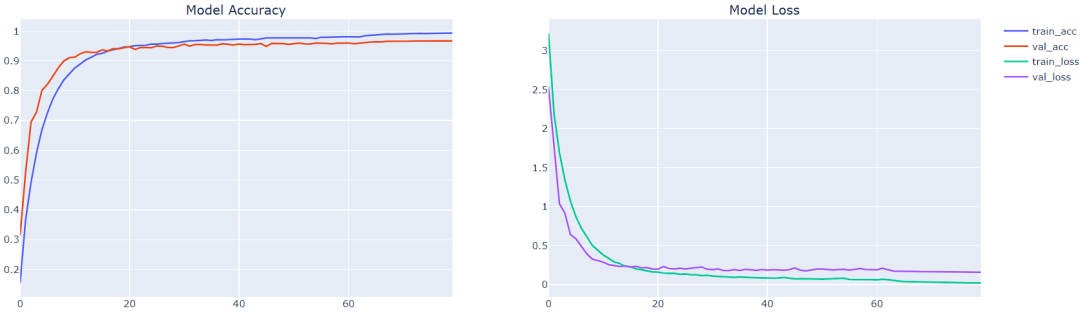

Figure 7.8: Proposed Model 3 Accuracy and Loss Graph

Index	precision	recall	f1-score	support
0	1.00	0.97	0.98	202
1	0.98	0.98	0.98	200
2	0.99	0.99	0.99	200
3	0.93	0.90	0.92	216
4	0.99	0.99	0.99	165
5	0.98	1.00	0.99	206
6	0.98	0.99	0.99	201
7	0.97	0.99	0.98	204
8	0.98	0.98	0.98	214
9	0.98	0.98	0.98	191
10	0.99	0.99	0.99	220
11	0.99	0.96	0.98	196
12	0.99	0.98	0.99	178
13	0.99	0.96	0.97	194
14	0.98	0.97	0.98	200
15	0.99	0.97	0.98	180
16	0.99	1.00	0.99	246
17	0.99	1.00	1.00	221
18	0.95	0.97	0.96	193
19	0.98	0.97	0.98	214
20	0.98	1.00	0.99	205
21	1.00	0.98	0.99	206
22	0.94	0.94	0.94	232
23	0.95	0.97	0.96	197
24	0.97	0.95	0.96	217
25	0.98	0.99	0.98	192
26	0.98	0.97	0.97	206
27	0.96	0.98	0.97	192
28	0.99	0.98	0.99	194
29	0.99	0.99	0.99	201
30	0.97	1.00	0.99	212
31	0.98	0.96	0.97	181
32	0.95	0.99	0.97	201
33	0.96	0.97	0.96	210
34	0.99	0.99	0.99	213
35	0.96	0.96	0.96	221
36	0.98	0.98	0.98	200
37	1.00	1.00	1.00	222
accuracy	None	None	0.98	7743
macro avg	0.98	0.98	0.98	7743
weighted avg	0.98	0.98	0.98	7743

Table 7.3: *Classification Report Model 1*

Index	precision	recall	f1-score	support
0	0.94	0.95	0.94	191
1	0.93	0.97	0.95	199
2	0.96	0.99	0.97	205
3	0.88	0.85	0.86	192
4	0.96	0.95	0.96	172
5	0.86	0.87	0.87	199
6	0.84	0.81	0.82	200
7	0.83	0.89	0.86	210
8	0.87	0.87	0.87	238
9	0.94	0.94	0.94	201
10	0.97	0.94	0.95	231
11	0.93	0.96	0.95	189
12	0.96	0.92	0.94	191
13	0.90	0.84	0.87	204
14	0.83	0.86	0.84	199
15	0.88	0.84	0.86	203
16	1.00	0.99	0.99	211
17	0.99	1.00	1.00	219
18	0.85	0.85	0.85	209
19	0.95	0.94	0.95	197
20	0.97	0.94	0.95	236
21	1.00	0.99	0.99	219
22	0.85	0.89	0.87	221
23	0.94	0.95	0.95	219
24	0.96	0.96	0.96	224
25	0.99	0.99	0.99	197
26	0.99	1.00	0.99	201
27	0.93	0.88	0.91	190
28	0.99	0.98	0.98	163
29	0.97	0.98	0.98	199
30	0.97	0.98	0.97	176
31	0.99	0.97	0.98	204
32	0.95	0.97	0.96	191
33	0.93	0.97	0.95	218
34	0.98	0.98	0.98	187
35	0.95	0.94	0.95	178
36	0.93	0.93	0.93	211
37	0.92	0.96	0.94	212
accuracy	None	None	0.93	7706
macro avg	0.93	0.93	0.93	7706
weighted avg	0.93	0.93	0.93	7706

Table 7.4: *Classification Report Model 2*

Index	precision	recall	f1-score	support
0	0.98	0.97	0.97	221
1	0.96	0.97	0.96	219
2	0.98	0.98	0.98	203
3	0.96	0.95	0.95	213
4	0.95	0.97	0.96	156
5	0.96	0.96	0.96	207
6	0.96	0.96	0.96	210
7	0.96	0.95	0.96	205
8	0.98	0.97	0.97	209
9	0.97	0.98	0.98	201
10	0.96	0.97	0.97	205
11	0.99	0.96	0.98	204
12	0.94	0.97	0.95	174
13	0.98	0.95	0.97	212
14	0.97	0.97	0.97	209
15	0.98	0.95	0.96	202
16	0.97	0.96	0.96	221
17	0.95	0.94	0.95	193
18	0.95	0.98	0.96	204
19	0.96	0.93	0.95	195
20	0.98	0.99	0.98	216
21	0.97	0.96	0.97	223
22	0.94	0.94	0.94	217
23	0.93	0.96	0.95	207
24	0.95	0.96	0.96	196
25	0.99	0.99	0.99	210
26	0.96	0.94	0.95	206
27	1.00	0.99	0.99	206
28	0.99	0.98	0.99	186
29	0.99	0.99	0.99	216
30	0.96	0.98	0.97	218
31	0.99	0.97	0.98	201
32	0.98	0.99	0.98	205
33	0.96	0.95	0.95	204
34	0.98	0.98	0.98	175
35	0.97	0.97	0.97	182
36	0.98	0.98	0.98	218
37	0.93	0.97	0.95	194
accuracy	None	None	0.97	7743
macro avg	0.97	0.97	0.97	7743
weighted avg	0.97	0.97	0.97	7743

Table 7.5: *Classification Report Model 3*

7.4 Ensemble Learning

There could be multiple approaches to ensemble techniques. The following passage is to describe all these techniques and finding the best possible solution for our domain-specific problem [7]. A ground truth label, n models, and n predictions from each model are needed for the ensemble model. The challenge is putting the best algorithm into practice so that the combination of n-predictions can give us matching ground truth. The proposed model_1, model_2, and model_3 are the models used for the ensemble's base learner. For the ensemble testing, the remaining 1520 training data are being used.

7.4.1 Unweighted Averaging Ensemble Technique

This is the most straightforward ensemble technique best suited for the problem. The method requires prediction for each sample from all three models. The ensemble prediction is created by adding the individual predictions from model 1, model 2, and model 3 for each sample. Then divide added probability by the number of models that are taking part. In our case, it would be 3.0 for having three different base learners. To further visualize the problem, assume a multi-class classification problem with four classes A, B, C, and D. Now, model 1, model 2, and model 3 are trained to classify an image into these categories. For each class, each model offers probability for each sample. Now for each sample, the maximum average of the probability of a class is the ensemble prediction.

Consider the probabilities below for a single sample where models 1, 2, and 3's predictions are [0.2, 0.3, 0.4, 0.1], [0.1, 0.5, 0.2, 0.2], and [0.3, 0.2, 0.1, 0.4]. Start with calculating the probability for each class. Add the probabilities together. The averaging approach divides the added probabilities by the number of models. So this would lead us to this series of equations: [((0.2 + 0.1 + 0.3) / 3.0), ((0.3 + 0.5 + 0.2) / 3.0), ((0.4 + 0.2 + 0.1) / 3.0), ((0.1 + 0.2 + 0.4) / 3.0)] or simplifying it to [0.2, 0.333, 0.233, 0.233]. The ensemble prediction probabilities are obtained by averaging the corresponding probabilities from each model. Then it needs to find the highest probability. In this case, the ensemble prediction probabilities suggest that class B is the predicted class as it has the highest probability.

7.4.2 Unweighted Voting Ensemble Technique

Assume there are 3 CNN models called Models 1,2 and 3. The models are trained to classify a sample into these four objects, A, B, C and D. When a sample is sent as input to the model, and the model provides a probability for each category. Table 7.6 shows the predicted class from each model. Table 7.6 suggests that Model 1 and Model 2 predict the outcome as Class C, while Model 3 predicts Class A.
Since Class C has the highest number of votes (2 votes), we choose Class C as the ensemble prediction. Therefore, in this case, where Model 1 and Model 2 predict the same class (Class C) while Model 3 predicts a different class (Class A), the voting ensemble technique would determine the ensemble prediction as Class C based on the majority votes.

Table 7.6

Model Index	Probability
Model 1	0.2, 0.3, 0.4, 0.1
Model 2	0.1, 0.1, 0.6, 0.2
Model 3	0.6, 0.2, 0.1, 0.1

Table 7.7

Class Index	Votes
Class A	1
Class B	0
Class C	2
Class D	0

7.4.3 Static-Weight Averaging Ensemble Technique

This ensemble technique, Weighted Average Ensemble, combines predictions from three base models (model1, model2, and model3) for a multi-class classification problem with classes A, B, C, and D. Each model provides probability predictions for each category. The individual forecasts for each sample from model1, model2, and model3 are combined using static weights derived from average precision values obtained during model evaluation.

Consider an example with the following probability predictions for a single sample in table 7.8

Table 7.8

Model Index	Probability
Model 1	0.2, 0.3, 0.4, 0.1
Model 2	0.1, 0.5, 0.2, 0.2
Model 3	0.3, 0.2, 0.1, 0.4

To obtain an ensemble prediction, we will calculate the average probability of each class considering the weighted sum of the probabilities. These weights correspond to accuracy values obtained in model1, model2, and model3 evaluation.

Equation 1 - Ensemble prediction probabilities for Class A:
[((0.2 * 0.98 + 0.1 * 0.93 + 0.3 * 0.97) / (0.98 + 0.93 + 0.97))]
Class A = 0.20138888888

Equation 2 - Ensemble prediction probabilities for Class B:
[((0.3 * 0.98 + 0.5 * 0.93 + 0.2 * 0.97) / (0.98 + 0.93 + 0.97))]
Class B = 0.33090277777

Equation 3 - Ensemble prediction probabilities for Class C:
[((0.4 * 0.98 + 0.2 * 0.93 + 0.1 * 0.97) / (0.98 + 0.93 + 0.97))]
Class C = 0.234375

Equation 4 - Ensemble prediction probabilities for Class D:
[((0.1 * 0.98 + 0.2 * 0.93 + 0.4 * 0.97) / (0.98 + 0.93 + 0.97))]
Class D = 0.23333333333

There is no complexity to the algorithm. Multiply the probability of a class by the model's weight (precision value). Then add all the probability and divide by

the models' total weight. Class B has the highest likelihood based on the ensemble prediction probabilities in this example. Therefore, the ensemble technique selects class B as the final prediction due to its highest probability among the classes A, B, C, and D.

7.4.4 Static-Weight Voting Ensemble Technique

This voting ensemble technique uses three base models (Model 1, Model 2, and Model 3) for multi-class classification. Each model provides probability predictions for each class. To incorporate the weights into the ensemble prediction, assign static weights to each model.

Let's assume the following weights on table 7.9 and predictions on table 7.10:

Table 7.9

Model Index	Weight
Model 1	0.98
Model 2	0.93
Model 3	0.97

Table 7.10

Model Index	Predictions
Model 1	Class C
Model 2	Class C
Model 3	Class A

Votes = [(Class C, 0.98), (Class C, 0.93), (Class A, 0.97)]
Sorted Votes = [(Class C, 0.98), (Class A, 0.97), (Class C, 0.93)]
Now, check if all models predict different classes. In this case, Model 1 and Model 2 predict Class C, while Model 3 predicts Class A. Since the models have no unanimous agreement, proceed with the majority voting approach. The ensemble prediction is obtained using the class with the most votes. In the example, Class C has two votes, while Class A has one vote. Therefore, the ensemble prediction would be Class C.
Assume a new prediction, votes and weights:

Table 7.11

Class	Vote
Class A	1
Class B	1
Class C	1
Class D	0

Table 7.12

Model	Predictions
Model 1	Class B
Model 2	Class C
Model 3	Class A

Table 7.13

Model	Weight
Model 1	0.98
Model 2	0.93
Model 3	0.97

Since all classes have an equal number of votes (1 vote each), there is no majority. In such cases, the approach is to consider the predicted class from the model with the highest weight as the ensemble prediction. Sorting the votes based on the weights in descending order:

Sorted Votes = [(Class B, 0.98), (Class A, 0.97), (Class C, 0.93)]

Now, choose the class predicted by the model with the highest weight, Class B. Therefore, the ensemble prediction in this case would be Class B.
The weights give more importance or influence to specific models in the voting process. The higher the weight assigned to a model, the more impact its prediction

has on the final ensemble prediction. In this case, Model 1 carries the highest weight (0.98), followed by Model 3 (0.97) and Model 2 (0.93). By incorporating the weights into the voting ensemble technique, we can effectively prioritize the predictions of specific models based on their performance or reliability, leading to an ensemble prediction that reflects the combined expertise of the individual models.

7.4.5 Dynamic-Weight Averaging Ensemble Technique

The process of the Dynamic-weight averaging technique is completely similar to the static-weight averaging ensemble technique, but as the name suggests, the weight is dynamic. For Model1, Model2 and Model3 we have the classification report table 7.3, 7.4 and 7.5. After getting the prediction from each model, take the precision of that specific class from the table and use that precision value as weight. Now For each class on each model, we have a dynamic weight that is not fixed.
Let's assume the static weights are as follows:

Weight for Model 1: 0.98
Weight for Model 2: 0.93
Weight for Model 3: 0.97

Assume Model 2 is really good at predicting class B with a precision of 1.0. However, the average precision falls behind because of the inability to predict the other class. This is where the dynamic weight can help; if Model 2 predicts Class B and the other two model predicts Class A and C, taking Model-2's prediction would make much more sense.

7.4.6 Ensemble Stacking Technique

The ensemble stacking technique is similar to all the work provided before. The change lies in the meta-learning model. Instead of using strict rules such as voting and averaging, meta-learning relies on a machine-learning algorithm for the ensemble prediction. The labels of the test datasets are then transformed into one-hot encoded format. Next, the test datasets and their corresponding labels are split into training and testing sets. Predictions are made on the training sets using the trained base models. These predictions are multiplied by respective weights assigned to each model. The weighted predictions are combined horizontally to create the training dataset for the meta-learner. A meta-learner is trained on the training dataset along with categorical labels obtained from the one-hot encoded labels. The meta-learner predicts the labels for the test dataset. Accuracy scores are calculated for each base model by comparing their predictions with the true labels, and the accuracy of the meta-learner is also calculated. Finally, the accuracy scores of the base models and the meta-learner are printed to assess the performance. For the meta-learner algorithms, we used Random Forest Classifier, Support vector machine, K-Nearest Neighbors, and Logistic Regression.
For meta-learner algorithms, we used Random Forest Classifier, Support vector machine, K-Nearest Neighbors and Logistic Regression.

7.5 Ensemble Modle Classification Report

The following report is based on the 1,520 testing samples we split earlier. Single model system refers to regular system where ensemble techniques are not being used. Different ensemble techniques are compared alongside how the regular model would perform independently. Firstly, by trying out state-of-the-art architectures to see how they would perform. By comparing Inception, DenseNet, Xception, Resnet50, and VGG19 on this validation set, it was noticed that Resnet50 outperformed all the models by a significant and convincing margin. On the other hand, a comparison between multiple ensemble techniques was studied. Starting with basic averaging and voting, then applying weight and dynamic weight to it, and finishing with ensemble stack modeling with different machine learning models. The study hypothesized that dynamic-weight-based averaging would provide the highest accuracy. Still, it was tested to be false as it came in third place only after the static-weight and unweighted averaging techniques. The voting and averaging models are similar in principle, but taking confidence into account were able to make a marginal difference in the total outcome. The ensemble stacking models worked well too, but they could not outperform the simpler decision-making algorithms.

Model	**Accuracy**
Inception (Single Model System)	0.7118
DenseNet (Single Model System)	0.7770
Xception (Single Model System)	0.6855
ResNet50 (Single Model System)	0.9349
VGG19 (Single Model System)	0.7822
Unweighted Averaging Ensemble Model	0.9500
Unweighted Voting Ensemble Model	0.9329
Static-Weight Averaging Ensemble Model	0.9513
Static-Weight Voting Ensemble Model	0.9454
Dynamic-Weight Averaging Ensemble Model	0.9480
Ensemble Stack RandomForest Meta-learner	0.9137
Ensemble Stack SVM Meta-learner	0.9194
Ensemble Stack KNN Meta-learner	0.9235
Ensemble Stack Logistic Regression Meta-learner	0.9367

Table 7.14: *Model Accuracy*

Chapter 8

Future Improvement and Discussion

8.1 Future Improvement

Our model is robust while detecting hand-sign, but there is always a place for improvement. Firstly our model is not hyperparameter tuned, so it is possible to reach new heights using the KerasTuner [52]. Secondly, a better dataset can improve the model significantly. A dataset with enough instances that require little augmentation, such as a thousand hand-sign samples for each class with different background colors, can provide better learning features. For having a vast amount of hand-sign based on white background, there are instances where the explanation for the hand sign is not appropriately justified for model 1. An interpretability algorithm LIME provides explanations for the predictions made by any classifier. It accomplishes this by locally approximating the target model with an interpretable model, ensuring trustworthy and understandable explanations for the predictions [18]. We used LIME or local interpretable model-agnostic explanation to visualize the superpixel that highlights the reasoning behind our hand sign.

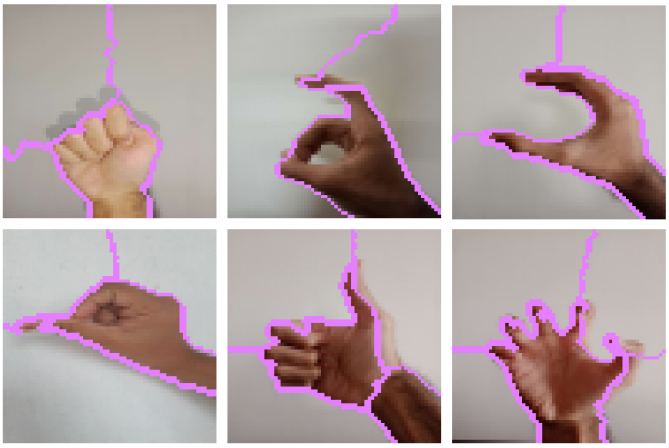

Figure 8.1: *Local Interpretable Model-agnostic Explanations report on superpixel segmentation of the image.*

From Fig: 8.1, the final photo represents the explanation for the prediction. The segmentation of the explanation can get even better if we can provide a better

dataset. The pre-processing technique for model 2 can still improve but giving each finger a different color. This procedure will allow our model 2 to understand the sample with more explanation. For instance, when the fingers overlap, providing a different color for each finger will allow Model 2 to track each finger and make some explanation for it.

8.2 Discussion

Our work focuses on developing new procedures and models with better outcomes. Combining multiple CNN models, we created a more robust system with better results and accuracy. While our study provides promising results, there are certain limitations. Our proposed procedure is highly dependent on the pre-processing technique and may require more time on pre-processing steps in real-life implementation. Secondly, our model 2 depends entirely on the MediaPipe library, so it may create a single point of failure for this model, even though accurate prediction from models 1 and 3 can overcome this problem entirely. We hypothesized that dynamic-weight averaging would provide the best result on the ensemble meta-learning algorithm, but it came with the third highest accuracy and very close to the best accuracy. More testing with different unseen data could justify the best meta-learning algorithm.

Detecting and identifying sign language is very important for establishing a more accessible communication network between people who are deaf and the rest of the world. New ideas are created through communication and knowledge, so sharing information is crucial for a better world. Communication has been a necessary tool since ancient times, and it is essential to allow speaking and letting those feelings, thoughts, and ideas be known to everyone. People who cannot communicate verbally have difficulty communicating with people who do not know sign language. Thus, a sign language recognition system is necessary. Using our model, we plan to create stronger connections between people with normal hearing and those with hard hearing.

Chapter 9

Conclusion

In this paper, we utilize three lightweight CNN models that can perform similarly to baseline models while using fewer resources and providing a prediction faster. The paper also compares our proposed CNN models with benchmark models such as Inception V3, ResNet50, Xception, and VGG19. Our proposed models came on top with better accuracy and prediction than the benchmark model on the same dataset, except for ResNet50 beating our model 2, which was trained on a different mode. The paper also helps to highlight challenges in generalization and dataset diversity in sign language and provides guidelines on how to tackle them. We aim to conduct further research on sign language recognition systems to provide a baseline of procedures.

Bibliography

[1] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[2] C. Vogler and D. Metaxas, "Parallel hidden markov models for american sign language recognition," in *Proceedings of the seventh IEEE international conference on computer vision*, IEEE, vol. 1, 1999, pp. 116–122.

[3] C. Goutte and É. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *European Conference on Information Retrieval*, 2005.

[4] Q. Munib, M. Habeeb, B. Takruri, and H. A. Al-Malik, "American sign language (asl) recognition based on hough transform and neural networks," *Expert systems with Applications*, vol. 32, no. 1, pp. 24–37, 2007.

[5] S. S. F. Begum and M. Hasanuzzaman, "Computer vision-based bangladeshi sign language recognition system," *2009 12th International Conference on Computers and Information Technology*, pp. 414–419, 2009.

[6] B. C. Karmokar, K. M. R. Alam, and M. K. Siddiquee, "Bangladeshi sign language recognition employing neural network ensemble," *International Journal of Computer Applications*, vol. 58, pp. 43–46, 2012.

[7] M. Ré and G. Valentini, "Ensemble methods : A review," 2012.

[8] B. Kang, S. Tripathi, and T. Q. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, IEEE, 2015, pp. 136–140.

[9] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv*, vol. abs/1511.08458, 2015.

[10] S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *ArXiv*, vol. abs/1503.06462, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:16159835.

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2015.

[12] K. H. Tarafder, N. Akhtar, M. M. Zaman, M. M. A. Rasel, M. M. R. Bhuiyan, and P. G. Datta, "Disabling hearing impairment in the bangladeshi population.," *The Journal of laryngology and otology*, vol. 129 2, pp. 126–35, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:25333292.

[13] S. T. Ahmed and M. A. H. Akhand, "Bangladeshi sign language recognition using fingertip position," *2016 International Conference on Medical Engineering, Health Informatics and Technology (MediTec)*, pp. 1–5, 2016.

[14] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2016.

[17] R. J. Johnson and J. E. Johnson, "Distinction between west bengal sign language and indian sign language based on statistical assessment," *Sign Language Studies*, vol. 16, pp. 473–499, 2016.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[19] M. A. Uddin and S. A. Chowdhury, "Hand sign language recognition for bangla alphabet using support vector machine," *2016 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pp. 1–4, 2016.

[20] S. Ameen and S. Vadera, "A convolutional neural network to classify american sign language fingerspelling from depth and colour images," *Expert Systems*, vol. 34, no. 3, e12197, 2017.

[21] U. Nations, *World population projected to reach 9.8 billion in 2050, and 11.2 billion in 2100 — UN DESA — United Nations Department of Economic and Social Affairs*, Jun. 2017. [Online]. Available: https://www.un.org/development/desa/en/news/population/world-population-prospects-2017.html.

[22] S. Yang and Q. Zhu, "Video-based chinese sign language recognition using convolutional neural network," in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, IEEE, 2017, pp. 929–934.

[23] K. Bantupalli and Y. Xie, "American sign language recognition using deep learning and computer vision," in *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, 2018, pp. 4896–4899.

[24] M. R. Islam, U. K. Mitu, R. A. Bhuiyan, and J. Shin, "Hand gesture feature extraction using deep convolutional neural network for recognizing american sign language," in *2018 4th International Conference on Frontiers of Signal Processing (ICFSP)*, IEEE, 2018, pp. 115–119.

[25] S. S. Shanta, S. T. Anwar, and M. R. Kabir, "Bangla sign language detection using sift and cnn," *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, 2018.

[26] S. Shivashankara and S. Srinath, "American sign language recognition system: An optimal approach," *International Journal of Image, Graphics and Signal Processing*, vol. 11, no. 8, p. 18, 2018.

[27] S. Ahmed, M. R. Islam, J. Hassan, *et al.*, "Hand sign to bangla speech: A deep learning in vision based system for recognizing hand sign digits and generating bangla speech," *ArXiv*, vol. abs/1901.05613, 2019.

[28] M. S. Islalm, M. M. Rahman, M. H. Rahman, M. Arifuzzaman, R. Sassi, and M. Aktaruzzaman, "Recognition bangla sign language using convolutional neural network," *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1–6, 2019.

[29] H. B. Nguyen and H. N. Do, "Deep learning for american sign language fingerspelling recognition system," in *2019 26th International Conference on Telecommunications (ICT)*, IEEE, 2019, pp. 314–318.

[30] A. M. Rafi, N. Nawal, N. S. N. Bayev, L. Nima, C. Shahnaz, and S. A. Fattah, "Image-based bengali sign language alphabet recognition for deaf and dumb community," *2019 IEEE Global Humanitarian Technology Conference (GHTC)*, pp. 1–7, 2019.

[31] M. M. Rahman, M. S. Islam, M. H. Rahman, R. Sassi, M. W. Rivolta, and M. Aktaruzzaman, "A new benchmark on american sign language recognition using convolutional neural network," in *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, IEEE, 2019, pp. 1–6.

[32] B. Shi, A. M. D. Rio, J. Keane, D. Brentari, G. Shakhnarovich, and K. Livescu, "Fingerspelling recognition in the wild with iterative visual attention," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5400–5409.

[33] J. Brownlee, *4 types of classification tasks in machine learning*, Aug. 2020. [Online]. Available: https://machinelearningmastery.com/types-of-classification-in-machine-learning/.

[34] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2020, pp. 1459–1469.

[35] A. Wadhawan and P. Kumar, "Deep learning-based sign language recognition system for static signs," *Neural computing and applications*, vol. 32, no. 12, pp. 7957–7968, 2020.

[36] T. A. Abedin, K. S. S. Prottoy, A. Moshruba, and S. B. Hakim, "Bangla sign language recognition using concatenated bdsl network," *ArXiv*, vol. abs/2107.11818, 2021.

[37] R. Dias, *American sign language hand gesture recognition*, May 2021. [Online]. Available: https://towardsdatascience.com/american-sign-language-hand-gesture-recognition-f1c4468fb177.

[38] M. A. Ganaie, M. Hu, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review," *ArXiv*, vol. abs/2104.02395, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:233033495.

[39] S. Jiang, B. Sun, L. Wang, Y. Bai, K. Li, and Y. Fu, "Skeleton aware multi-modal sign language recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3413–3423.

[40] S. Jiang, B. Sun, L. Wang, Y. Bai, K. Li, and Y. R. Fu, "Sign language recognition via skeleton-aware multi-model ensemble," *ArXiv*, vol. abs/2110.06161, 2021.

[41] R. A. Nihal, S. Rahman, N. M. Broti, and S. Ahmed Deowan, "Bangla sign alphabet recognition with zero-shot and transfer learning," *Pattern Recognition Letters*, vol. 150, pp. 84–93, 2021, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2021.06.020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865521002269.

[42] F. B. Slimane and M. Bouguessa, "Context matters: Self-attention for sign language recognition," *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 7884–7891, 2021.

[43] A. Hasib, S. S. Khan, J. F. Eva, *et al.*, "Bdsl 49: A comprehensive dataset of bangla sign language," *ArXiv*, vol. abs/2208.06827, 2022.

[44] P. D. Moral, S. Nowaczyk, and S. Pashami, "Why is multiclass classification hard?" *IEEE Access*, vol. 10, pp. 80 448–80 462, 2022. DOI: 10.1109/ACCESS.2022.3192514.

[45] N. E. Ramli, Z. R. Yahya, and N. A. Said, "Confusion matrix as performance measure for corner detectors," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 2022.

[46] A. Sen, T. K. Mishra, and R. Dash, "A novel hand gesture detection and recognition system based on ensemble-based convolutional neural network," *Multimedia Tools and Applications*, pp. 1–24, 2022.

[47] W. H. O. Africa, *Ear Health*, Jul. 2023. [Online]. Available: https://www.afro.who.int/health-topics/ear-health.

[48] *Hand landmarks detection guide 2023*, Jul. 2023. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker.

[49] M. S. Islam, A. Joha, M. N. Hossain, S. Abdullah, I. Elwarfalli, and M. M. Hasan, "Word level bangla sign language dataset for continuous bsl recognition," 2023.

[50] W. H. O. WHO, "Deafness and hearing loss," *World Health Organization*, Feb. 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

[51] *Opencv*. [Online]. Available: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html.

[52] K. Team, *Keras documentation: Kerastuner*. [Online]. Available: https://keras.io/keras_tuner/.