BDSLI: A Hybrid CNN-Transformer Model for Bengali Sign Language Interpretation

by

Abir Bin Yousuf
22366006

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

Department of Computer Science and Engineering
BRAC University
March 2025

Declaration

It is hereby declared that

1. The thesis submitted is my own original work while completing M.Sc. in Computer Science and Engineering 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 references.
3. The thesis does not contain material that has been accepted or submitted for any other degree or diploma at a university or other institution.
4. I have acknowledged all main sources of help.

Student's Full Name & Signature:

Abir Bin Yousuf
22366006

Approval

The thesis titled "BDSLI: A Hybrid CNN-Transformer Model for Bengali Sign Language Interpretation" submitted by
Abir Bin Yousuf (22366006)
Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science and Engineering on March 25, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
BRAC University

Examiner:
(External)

Dr. Mohammad Shafiul Alam
Professor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology (AUST)

Examiner:
(Internal)

Mr. Moin Mostakim
Senior Lecturer
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. Md Sadek Ferdous
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Ethics Statement (Optional)

This study strictly follows the guidelines to ensure that the data is collected, used and stored in a responsible manner. The dataset used in this research consists of video recordings of individuals performing Bengali Sign Language gestures. All participants provided their informed consent prior to data collection and their identities were anonymized to protect privacy. The study was carried out in accordance with the ethical guidelines of BRAC University for research involving human subjects. No personally identifiable information was stored or shared, and data usage was limited strictly to academic and research purposes. Any potential biases in the dataset were carefully considered to minimize unfair representations.

Abstract

Sign Language Recognition (SLR) is a widely explored field of study all over the world, yet progress remains limited for certain languages, including Bengali. In this study, a hybrid model (CNN + Transformers) is proposed to recognize isolated Bengali sign words and generate meaningful sentences. To the best of our knowledge, this specific combination has not been explored before. As a prerequisite for training the model, a video dataset is constructed comprising 62 Bengali sign words, with 250 videos per class. In addition, a separate test dataset is created to evaluate the model performance on unseen data. Additional hybrid models, including CNN+LSTM, CNN+BiLSTM, and CNN+GRU, are trained and evaluated alongside the proposed approach. Furthermore, widely recognized architectures such as LSTM, GRU, TCN, and Transformers are also implemented to demonstrate the superiority of the proposed model for the training dataset. In this study, it will be demonstrated that the chosen model can achieve an impressive 99.58% accuracy (with 99.48% validation accuracy) for the training dataset and 98.65% accuracy for the test dataset. These results surpass those of all other models that are trained and evaluated in this study. Following a comprehensive evaluation, this research intends to deploy the trained model in a web application to illustrate its effective performance.

Keywords: SLR, CNN, Dataset Construction, Transformers, Model Evaluation, Model Deployment

Dedication (Optional)

This research paper is devoted to the people who supported me, inspired me, and sincerely believed I could finish my thesis. I will forever be grateful to them.

Acknowledgement

Firstly, all praise to the Great Allah for whom my thesis has been completed without any major interruption.
Secondly, thanks to my supervisor Dr. Muhammad Iqbal Hossain Sir for his kind support and advice in my work. He helped me whenever I needed guidance.
Thirdly, thanks to the esteemed professors who attended the thesis panel and gave me insightful feedback that helped me proceed on the correct track.
Finally, thanks to my parents and friends as without their continuous support, it may not have been possible. With their kind support and prayer, I am now on the verge of completing my M.Sc. in Computer Science and Engineering degree.

Table of Contents

List of Figures

List of Tables

Nomenclature

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

$BDSL$ Bengali Sign Language

$BiLSTM$ Bidirectional Long Short Term Memory

CNN Convolutional Neural Network

GRU Gated Recurrent Units

$LSTM$ Long Short Term Memory

SLR Sign Language Recognition

SLT Sign Language Translation

TSL Thai Sign Language

UN United Nations

WFD World Federation of the Deaf

Chapter 1

Introduction

1.1 Motivation

According to an article [26] published by the United Nations, World Federation of the Deaf (WFD) claims that sign language is used as the only form of communication by 70 million people around the world. Unfortunately, most able-bodied people have little to no idea about sign language. Some people find it challenging to learn new languages, while others do not consider the sign language important enough to learn. This research paper is aimed at those people who want to communicate with the deaf community without having to learn a language from scratch. Some individuals even make the effort to learn sign language; however, they forget most of the signs because they do not use sign language regularly. In this competitive world, most people do not have enough time to learn something new. Therefore, researchers have been working with sign language recognition models to reduce their burden. We are also determined to do something for the community of deaf and mute people. Everyone will have the ability to converse with the members of that community with ease if a model can be suggested that is capable of translating and interpreting sign words and full sign sentences. An app built around such a model will definitely make disabled people more comfortable using sign language while talking to others. In this research, it is our aim to work towards figuring out the model which can yield the greatest performance for the mentioned function and deploying the model in a Web application.
In an article [26] by the United Nations (UN), it is stated that the deaf and mute community throughout the world are using more than 300 different types of sign language. Among them, the most recognized sign languages are ASL and BSL which represent American sign language and British sign language respectively. Numerous research studies have been conducted on the development and proposal of models capable of recognizing sign alphabets and sign words from ASL and BSL. Some of our senior researchers from Bangladesh have also worked with BDSL which is short for Bengali Sign Language. Unfortunately, progress for BDSL is comparatively slower than that for ASL and BSL. We have come across some work on word-level prediction, but we have failed to find any papers on BDSL with sentence-level recognition models. Observing that, we have decided to work on a model that will be able to recognize different words in real time and produce meaningful sentences as output using the predicted words. To be more specific, this study aims to build an app that can be used to have a face-to-face conversation with deaf individuals

who use sign language. Their signs will be captured with the camera on the user's smartphone, and the captured signs will be recognized to produce Bengali sentences as output.

1.2 Research Gap and Contribution

When we looked at research papers on this area of study, we noticed that some models are used frequently, including Hidden Markov Models, LSTM, CNN + LSTM, GRU, etc. After reviewing research papers that worked with word-level and sentence-level sign language recognition, we decided to work with a new model, which has not yet been used for sign language recognition. We have tried some popular object detection models such as YOLOV8, Mobilenetv2 and RandomForestClassifier. For these models, we had to collect an image dataset. However, we realized that images are not suitable for sign language recognition because sometimes there are multiple hand gestures for one sign. So, it was decided that we will use a more complex model that can be trained with a video dataset. After that, we looked into some complex models that can yield good results even for complicated hand signs.

After a thorough research, the primary model we agreed to use for our BDSLI app is a hybrid model architecture that combines CNNs and Transformers. The convolutional layers present in the CNN architecture are perfect for identifying local patterns. Moreover, CNN's MaxPooling layers reduce the dimensionality while keeping the most relevant features. On the other side, the multi-head attention layer in Transformers captures global dependencies and global patterns from the input data. This is especially helpful for sequential datasets where feature interaction and position are important. Additionally, Layer Normalization ensures better training stability, which is critical for deep learning models. We also discovered that in many fields, the combination of CNN (for feature extraction) and Transformers (for sequence relationships) provides a revolutionary and effective option that performs better than either CNN or Transformer models separately, which is why we were determined to use this hybrid model.

Later, we experimented with different versions of this model and trained them on different versions of our dataset. As there was no notable video dataset available for the Bengali sign language, we had to build a dataset of our own from scratch. Collecting an appropriate dataset for our model took some time. After choosing the final version of the dataset, we trained our CNN+Transformers model. In the following step, we also trained 7 other models on the same dataset to see whether our chosen model has the best training and validation accuracy. Reviewing the results of the comparison, we decided to proceed with our model as it had the highest accuracy. After that, we thoroughly compared our model with previously used models with a test dataset. We compared our model with models including GRU, LSTM, CNN+LSTM, TCN, Transformers, CNN+GRU and CNN+BiLSTM. Fortunately, CNN+Transformers managed to achieve the highest test accuracy among the mentioned models. After deciding on the final version, we deployed it into a Web App that can interpret Bengali sign language live from the webcam and show complete, meaningful sentences as output to the user.

1.3 Research Questions

This research will focus on answering the following questions related to this field of study:

- Do hybrid models perform better than standalone models for Sign Language Translation and Recognition?
- Can the proposed hybrid architecture outperform the models which were introduced in the previous research papers?
- Does the proposed model yield good results for both seen and unseen data?
- Is it possible to train the hybrid model on words that contain dynamic gestures and can it recognize those words accurately?

1.4 Aims and Objectives

Developing a sign language recognition model requires learning the sign language or understanding its usage. Before starting to work on this thesis, our idea about Bengali sign language was insubstantial, and we had to watch several videos on YouTube to decide which words can be chosen for this study. Here, we want to give credit to some YouTube channels, including Grameenphone and Robi, for teaching us signs for a large number of words which are used frequently in a day-to-day conversation. We started our study with the following seven clear objectives in mind.

1. At first, we want to select some frequently used words to create a video dataset for this study. An accurately designed dataset is pivotal to train a model properly.
2. In the next step, we need to extract some relevant features from our training dataset and store those extracted features in a pickle file. This pickle file will be used to train our model.
3. Our third objective is to train our CNN+Transformer model with the pickle file and observe the training and validation accuracy.
4. In the subsequent step, we want to train 7 other models with the same pickle file we created by extracting features from our dataset. After that, we compare the accuracy values (precision values, training scores, validation scores, recall values, and F1 scores) of the trained models with our primary model.
5. The fifth objective is to check whether our model is capable of predicting the sign language live from the webcam with an appropriate prediction code.
6. The next step is to create a test dataset that will be used to evaluate the models on unseen data. Then, we have to analyze the results to see if the proposed model prevails.

7. The last objective is to deploy the final version of our model into a Web App and make sure the Web App is capable of interpreting Bengali sign language in real time.

Throughout this research paper, the aforementioned objectives will be met with sufficient explanation and supporting information.

1.5 Outline of the Thesis

The structure of this research study is as follows:

Chapter 2: This chapter discusses some earlier studies on sign language recognition and translation. Several models which were used in different research papers are discussed here. The limitations of those papers are also mentioned along with the improvements that can be made.

Chapter 3: This chapter covers the proposed method, which includes detailed information on the training and test dataset. Accumulation of data and data preprocessing, two different techniques for extracting features (FE techniques), and an overview of the hybrid model are also covered here with some relevant figures and equations.

Chapter 4: In Chapter 4, the entire training process is described with an in-depth demonstration of the hybrid model's architecture. Different line graphs are added to depict how the model performs throughout the training process. In the last section of this chapter, the trained hybrid model is evaluated on a separate test dataset and a confusion matrix shows how accurate the model is in recognizing each word class.

Chapter 5: In the first segment of this chapter, seven models are trained on the same dataset. The following segment compares the training data of the models using two different tables. Then, the models are tested on test dataset and the test results are compared. After that, it is shown how the proposed hybrid model interprets Bengali sign words and constructs full sentences with those words. The last segment of the chapter compares the proposed model with some of the models introduced in previous research papers.

Chapter 6: This is the last chapter that concludes this research paper by summarizing the contributions we tried to make to this field of study. This chapter also shows the limitations of our model and what we want to do in the future to overcome them. Lastly, an innovative future experiment is mentioned, that has the potential to be truly beneficial to society.

Chapter 2

Related Work

Sign language interpretation is a heavily explored field of study as a large number of researchers are attempting to determine the most effective method for this task. We have come across a large number of research papers on ASL, BSL and other widely used sign languages. However, Bengali sign language (BDSL) is not as explored as other languages. There are some studies on BDSL but they do not work on sentence-level interpretation. In the research paper [9], Tazalli et al. 2022 worked with isolated SLR where isolated sign words are detected from videos. For this study, they used YOLOV5 and PyTorch to build their model and trained the model with a dataset consisting of videos for 34 different words. The problem with their approach was that YOLOV5 is a model that does not work well for sign words that have complicated hand gestures. As a result, their accuracy did not improve as expected. Moreover, they did not predict full sentences as outputs which we are trying to achieve with this research. It is also our plan to use a hybrid complex model which is capable of accurately interpreting complicated hand gestures. In addition, we want to build a bigger dataset with 62 words and significantly more videos for each sign word.

In a recent research paper [22], Zhang et al. 2024 suggested a CNN oriented model which incorporates a dual-path background erasure technique to learn features. Their proposed model was trained on a dataset with 24 different classes with 500 videos for each class. The trained model achieved 99.52% accuracy with a macro F1-score of 0.997. However, this particular model had some limitations as it was only trained to recognize alphabets, and was unable to recognize full words. Another limitation of this model was that it could not process dynamic gestures which is why the alphabets 'J' and 'Z' were excluded from the dataset. We are trying to overcome these limitations and we want to make sure that the model we will show in this paper will not only be able to recognize dynamic gestures, but also interpret full-length meaningful words. This paper will also try to match if not surpass the exceptional accuracy obtained in the mentioned research paper [22].

We took inspiration from the research papers [22], [16] and [2] to use CNN as our main algorithm as these papers showed that CNN can perform well in this domain of research. Unfortunately, all of these papers worked with alphabet recognition and the models trained for these papers were unable to interpret sign words. It is already established that CNN is the state-of-the-art algorithm for the tasks related

to object detection. In the research work [16], Gayathri D. et al. 2024 incorporated a pre-trained Mobile-Net V2 architecture with CNN to build their model. Despite the fact that their model achieved 85.45% accuracy, the model was trained on a small dataset which is a noteworthy limitation. Similarly, the next research paper [2] written by Halvardsson et al. 2020 talks about using the InceptionV3 model with CNN which is a famous model for object detection from images. Both of these papers have the same problem which is that the models introduced in these papers only work with images that contain simple gestures. They cannot comprehend or translate complicated and dynamic gestures.

A notable research paper we have read was [17], where Kumari et al. 2024 worked with a hybrid CNN+LSTM model combined with MobileNetV2 and trained this model with the benchmark WLASL dataset. This suggested architecture attained 84.65% accuracy which is a noteworthy achievement on their part. However, they worked with word level recognition and their model was not programmed to construct sentences with the predicted words. Moreover, the accuracy needed to be increased to make the model stand out more. In another paper [4], Chaikaew et al. 2021 claimed that the model they proposed based on Recurrent Neural Networks (RNN) can accurately recognize sign words just like LSTM, BiLSTM and GRU models. In their paper, they worked with Thai Sign Language (TSL) and used mediapipe to extract hand landmarks from the dataset. We took inspiration from this research paper to use mediapipe, while trying to considerably increase the accuracy.

The research paper [19] by Paul et al. 2023 discusses several approaches for sign language recognition including a ResNet-based CNN model, LSTM and GRU. The researchers used the same dataset to train these three different models and showed that LSTM can outperform both ResNet and GRU with a big difference in performance. However, these models were only trained on 3 words and 26 alphabets. Hence, predicting full-length meaningful sentences was not possible for these trained models. Another exception research paper was [20] which was presented by Rao et al. 2024. In this detailed study, the researchers covered the development of the models that have been applied for SLR and SLT. Several models were mentioned in this paper including ConvLSTM, BiLSTM, BAE, BERT, SignBERT+, H-GAN, GCN and several others. From their study, we realized that there are plenty of studies on isolated sign words recognition, whereas sentence level detection is almost non-existent.

Another interesting research paper we came across was [14], where Xu et al. 2023 worked with a selective kernel-based TCN model (Temporal Convolutional Networks model) and the model was named SKResNet-TCN. This model was proposed to recognize sign words from videos and it outperformed both LSTM and 3DCNN. The researchers trained this SKResNet-TCN model on the Argentine LSA64 dataset and the model achieved a training accuracy of 100%. We are trying to come up with a hybrid model that can yield better results than this modified TCN model. Another new approach was seen in the research paper [1] where Camgoz et al. 2020 proposed transformers based models for sign language recognition and showed that transformers based models can achieve better results compared to other models. In the research work [8], Ma et al. 2022 introduced 4 two-stream mixed CNN models and proved that TSM-ResNet50 is the model with the best performance. The

researchers were able to attain an accuracy of 97.57% when they trained the TSM-ResNet50 model separately on MNIST dataset and ASL dataset.

Another research paper we want to mention is [12], where R.Sreemathy et al. 2023 worked with two different models for word-level sign language recognition. In their paper, they trained SVM and YOLOV4 on a dataset with 80 different word classes. Despite having a large number of classes, the models would not support dynamic gestures as they were trained on static images of words. Lastly, Rahman et al. 2021 proposed Generative Adversarial Networks (GANs) for word-level SLR. In their paper [6], they have shown that their model can achieve 91.3% accuracy after being trained on a dataset which contains 20 ASL words. We are trying to surpass the accuracy achieved in all of these previous papers with our hybrid CNN+Transformers model, which will be capable of dealing with the constraints the mentioned research papers faced.

Chapter 3

Proposed Method

3.1 Data Collection

Training Dataset: At first, we watched several tutorial videos on Bengali sign language to determine the specific words we want to use for our dataset. From hundreds of words, we narrowed it down to 62 words that are often used in our daily conversations. We started to record videos to build our dataset and we recorded them using our webcam. Before recording the videos, we had to minutely learn the signs for those selected 62 words. As Bengali words are not supported in LaTex, we have shown the words in Figure 3.1. In order to capture videos from webcam and

আজ, আবার, আমাদের, আমার, আমি, আসা, কবে, কয়টা বাজে,
কাজ, কিভাবে, কী, কে, কেন, কোথায়, খাওয়া, ছেলে, ডাকা, তাদের,
তুমি, তোমাদের, দয়া করে, দুঃখিত, দুপুর, দেখতে, দেখা, দেখা হয়ে,
দেশ, ধন্যবাদ, না, নাম, পড়া, বন্ধু, বলা, বাংলা, বাড়ী, বাবা, বুধবার,
বৃহস্পতিবার, বোন, ভাই, ভালো, ভাষা, মঙ্গলবার, মা, মেয়ে, যাওয়া,
রবিবার, রাত, লেখা, শনিবার, শুক্রবার, শোয়া, সকাল, সাহায্য, সিনেমা,
সুন্দর, সে, সোমবার, স্ত্রী, স্বামী, হবে, হ্যা

Figure 3.1: Selected 62 Bengali words for constructing training dataset

save them to their designated directories, we had to use two Python libraries which are cv2 and os. We collected 250 videos for each word which sums up to 15500 videos in total. The sequence length of each video was 30 which means a video had 30 frames in it. We tried to increase it to 60, however, it would have decreased the accuracy because we wanted real time prediction from our model. Hence, we agreed to keep the sequence length at 30. Figure 3.2 shows the screenshots taken from the sample videos of our dataset. The picture in the upper-left corner on Figure 3.2 depicts the sign for the word 'Ami', image in the top-right corner represents the sign for 'Dekha'. The remaining pictures represent the Bengali words 'Baari', 'Ha', 'Shokal' and 'Shahajjo' respectively. These captured videos are kept in 62 different folders and all of the folders are kept in a folder named dataset. This was used to extract features from the videos and train the model on the features. A small part of the training dataset was also used as the validation dataset when the model was being trained.

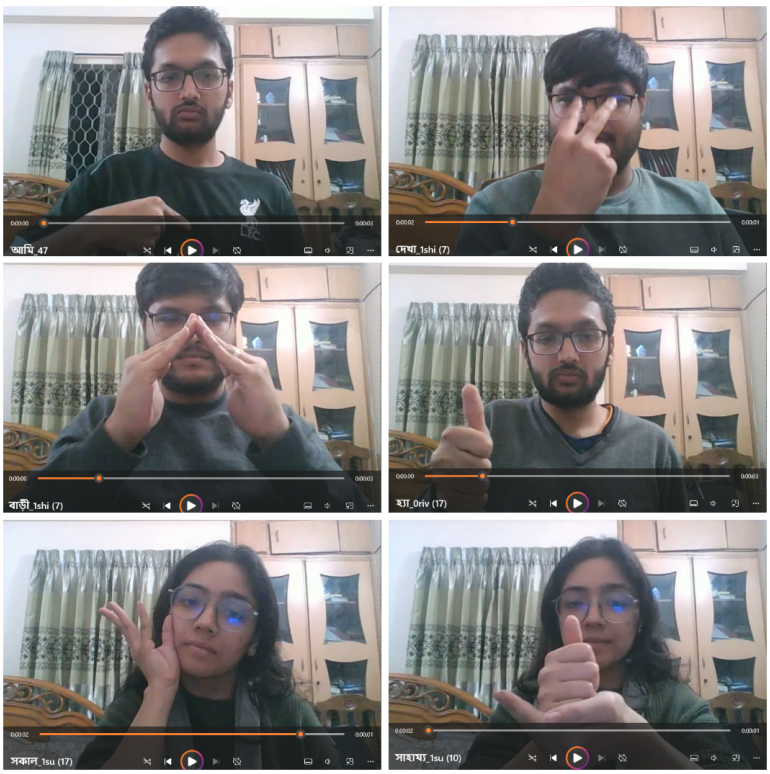

Figure 3.2: Screenshots from videos representing the words 'Ami', 'Dekha', 'Baari', 'Ha', 'Shokal' and 'Shahajjo'

Test Dataset: For constructing the test dataset, we had to keep the same words as training dataset. However, the videos needed to be captured again so that we can get a dataset which is different from the training dataset. This is to ensure that we have some unseen data to find out whether our model can predict Bengali sign language words from these unseen videos. Here, we captured 3100 videos with 50 videos per word. We kept the sequence length at 30, just like the training dataset. A video is 3 seconds in length which means it took the webcam 3 seconds to capture 30 frames. As we are working with real time prediction here, the model should not take more than 1 second to predict a word from a given sign. So, one might think why the webcam is taking longer to capture the whole sequence. The key consideration here is that the videos in the dataset have been intentionally slowed to facilitate accurate feature extraction. In reality, the actual duration of each video is only one second. So, our model was accurately extracting features from 30 frames and assigning appropriate class labels to those features, which is how the model learns different Bengali sign words.

3.2 Data Preprocessing

Initially, we planned to have only 50 videos per word class for the training dataset and 5 videos per word class for the test dataset. We did it because we wanted to save time and computational power. However, the dataset was too small and as a result, the model was not performing well. Then, we decided to increase the number of videos and started capturing them and saving them in their designated

folders. After we captured all the videos for both training and test dataset, we had to examine all the videos thoroughly. Firstly, we needed to make sure each video is kept on the correct folder. Then, we had to check the clarity of the videos because if the videos were unclear and blurry, we would not be able to extract features from them. If we did not extract accurate features, we would be unable to train our model properly. Whenever we came across blurry videos in the dataset, we replaced them by capturing new ones. In this way, we made sure that we have a remarkable dataset without any redundant data. We also had to ensure that there were 250 videos in all the folders representing different words in the training dataset. Similarly, the test dataset was also checked to see if each word folder contained 50 videos. During feature extraction, a small number of videos were discarded as the videos had inconsistent shape. For example, 41 videos from test dataset were removed and features were extracted from 3059 videos instead of 3100. Similarly, 209 videos from the training dataset were discarded for having inconsistent shape and features were extracted from 15291 videos.

3.3 Feature Extraction Techniques

3.3.1 Hand Landmarks

Before we could train our model, some features needed to be extracted from the videos. As we were working with sign language where hand gestures are the most important things, we decided to extract hand landmarks from our dataset. A Python Library called mediapipe is used to extract hand landmarks from each video. The hand landmarks show us the posture of the hands and landmarks are drawn differently for non-identical gestures. Figure 3.3 illustrates some of the landmarks extracted for a small number of words. Previously, we have shown some of the

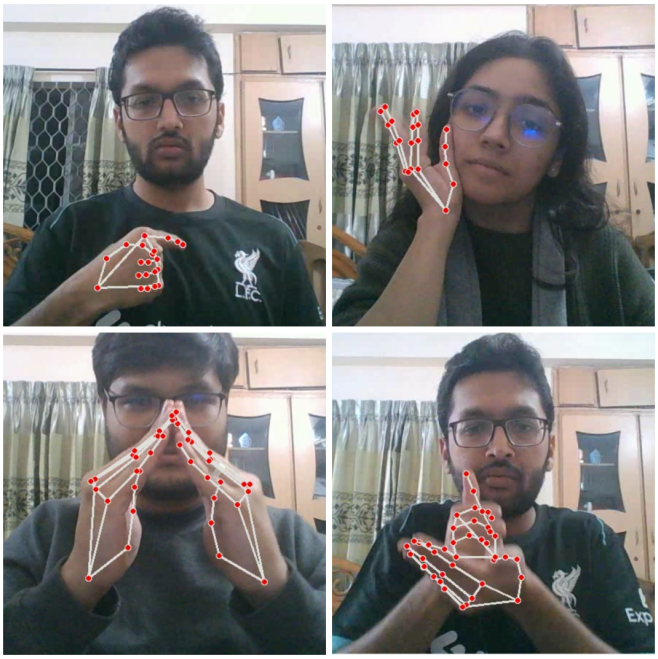

Figure 3.3: Extracted hand landmarks for the words 'Ami', 'Shokal', 'Baari' and 'Shahajjo'

words from our dataset including the words 'Ami', 'Shokal', 'Baari' and 'Shahajjo'. From the sign videos that represent these words, the feature extraction code extracted the hand landmarks demonstrated in Figure 3.3. These dissimilar landmarks are labeled appropriately and saved in a pickle file. We also needed to import several other libraries such as os, cv2, numpy and pickle to make the operations possible. Now, each hand holds 21 landmarks in total and each landmark has 3 coordinates x, y and z. Therefore, each hand would give us 21*3=63 features. The feature extraction technique will label these 63 features with the appropriate word class. This is how our model will be able to separate one word from another. We were using the saved pickle file to train our model and it was giving us good training and validation accuracy. It took around 4 hours to extract hand landmarks from the training dataset that contained 15500 videos. For the test dataset it only took more than an hour as the test dataset only had 3100 videos.

3.3.2 Hand Landmarks with relative position

Although, we were getting good training and validation accuracy with only hand landmarks, our model was getting confused with similar words. For example, the hand gesture for the words 'Maa' and 'Shukrobar' are almost the same as depicted with the two rightmost pictures in Figure 3.4. As a result, the hand landmarks were also similar for these two words. For that reason, our model was getting confused with these words and giving wrong predictions. In order to deal with this problem, we decided to add one extra feature for every gesture. This extra feature shows the exact position of a hand in the video frame. To be more precise, this is the average position (center) of a hand. This feature also tracks the hand in the video frame and changes the center when the hand is being moved around the frame. The problem with similar hand gestures was fixed with this feature. Here, we extract features from hand landmarks using the MediaPipe Hands model. Each hand consists of 21 landmarks and each landmark provides three spatial coordinates: (x, y, z). The extracted features consist of:

1. **Landmark Coordinates Extraction:** Each detected hand provides 21 landmarks, and each landmark i has three coordinates (x_i, y_i, z_i). The feature vector of landmarks is represented as mentioned in the website [25]:

$$F_{\text{landmarks}} = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_{21}, y_{21}, z_{21})\} \tag{3.1}$$

   This results in a feature vector of size $21 \times 3 = 63$.

2. **Hand Position Calculation:** To capture the overall position of the hand, we compute the centroid of the 21 landmarks using the mean of their coordinates:

$$x_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3.2}$$

$$y_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{3.3}$$

$$z_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} z_i \tag{3.4}$$

In the provided equation, N represents the total number of hand landmarks and $N = 21$ as stated in the book [15] by Bhatti et al. 2024. These three values $(x_{\text{avg}}, y_{\text{avg}}, z_{\text{avg}})$ are added to the feature vector.

3. **Final Feature Vector:** The final feature vector for a single frame consists of the landmark coordinates along with the average hand position:

$$F = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_{\text{avg}}, y_{\text{avg}}, z_{\text{avg}}] \tag{3.5}$$

This results in a total of $63 + 3 = 66$ features per frame.

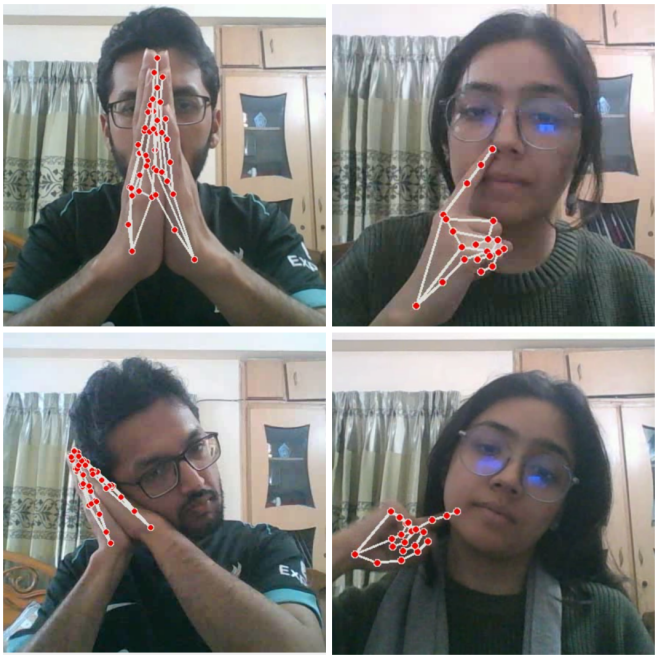

Figure 3.4: Similar hand landmarks for 'Doya kore' and 'Showa', 'Maa' and 'Shukrobar'

If we closely look at the left part of Figure 3.4, we can see that the hand gestures for 'Doya kore' and 'Showa' appear identical. So, when the hand landmarks are extracted from these gestures, they also looked almost the same. Similarly, the landmarks for the words 'Maa' and 'Shukrobar' are also indistinguishable. These indistinct words resulted in the model making the wrong prediction from time to time. We had to introduce the new feature to solve this issue. If we look at Figure 3.4 again, we will see that the hands have different positions. The relaive position (center) of the hands for 'Doya kore' are exactly in the middle of the video frame, whereas the center for 'Showa' is slightly bended on the left side of the frame.

The differences in the positions of hands are taken into account and used as a new feature which solves the similarity issue for us. It took more than 4.25 hours to extract hand landmarks with their relative position from the training dataset. For the test dataset, it took around 1.25 hours.

3.4 Hybrid (CNN+Transformers) Model

3.4.1 Convolutional Neural Network (CNN)

CNN is one of the most popular deep learning models which was especially designed to deal with structured data such as image and video datasets. CNN applies convolutional operations for extracting spatial and hierarchical features automatically from the input data as stated in the research paper [21] by Richhariya (2024). This is what makes CNN one of the most suitable models for computer vision tasks. Unlike traditional fully connected networks, CNN utilizes weight-sharing mechanisms in the form of filters (kernels) to learn local patterns and reduce the number of parameters. As a result, the computational efficiency of the model is increased.
The focal components of the CNN architecture are as follows:

1. **Convolutional Layers**: The key task carried by these specific layers is to perform convolutional operations on the input sequence using kernels.These filters (kernels) go through the given data and try to capture local features including shape, texture and edge. Deeper convolutional layers capture higher-level abstractions by combining low-level features.

2. **Pooling Layers**: These specific layers are also called downsampling layers. Such layers carry out the task of decreasing the spatial dimensions of the feature maps as mentioned in an article [24]. However, the most significant features are preserved by the pooling layers when such operations take place. Among the pooling techniques, average pooling and maxpooling are used regularly. These techniques aid in mitigating overfitting and also lower the complexity of the training process.

3. **Activation Functions**: These functions make the network non linear so that the model becomes capable of capturing complex features as well as simple ones. According to the book [7] by Tyagi (2021), ReLU which stands for Rectified Linear Unit is the most widely applied activation function in this architecture. This activation function is popular because it can deal with critical issues such as vanishing gradient.

4. **Dense Layers/ Fully Connected Layers**: When convolutional layers complete extracting features, dense layers accumulate the extracted information to make final predictions. These fully connected layers then enable the model to perform classification or regression tasks by connecting the output neurons with the neurons from the previous layers.

5. **Batch Normalization and Dropout**: Batch normalization normalizes activations within each batch to stabilize training and improve convergence.

Dropout is used to mitigate overfitting when the model is being trained. According to Ghayoumi (2023), dropout is a regularization technique where some neurons are disabled to prevent the model from overfitting [10].

It is already established that CNN is the most popular architecture or model for computer vision applications such as detecting objects and image/video classification. CNN excels at these tasks because it has the ability to learn spatial patterns from the image or video dataset autonomously.

3.4.2 Transformers

Transformers were especially developed for tasks related to natural language processing (NLP). This deep learning model (architecture) has revolutionized the field of NLP. However, this architecture has also been successfully adapted for vision tasks. Unlike CNN, which relies on local receptive fields, Transformers use self-attention mechanisms to capture long-range dependencies and global context in the input data.
The core components of the Transformers architecture are as follows:

1. **Self-Attention Mechanism**: This mechanism enables transformers to assign attention scores for each input element in relation to every other element in the input sequence. With this particular mechanism, transformers figure out which elements are more significant compared to others. In this way, the model can efficiently choose the important long-range patterns.

2. **Positional Encoding**: Unlike Recurrent Neural Networks (RNNs), Transformers do not handle data sequentially. As transformers use multi-head attention mechanism, the input data is handled in a parallel manner. In order to make such operations possible, the actual order of the input features need to be retained. This is where positional encoding is utilized and each input element is assigned with a positional encoding. This type of encoding contains functions of sin and cos for creating unique representations for different positions in a sequence.

3. **Multi-Head Attention Mechanism**: Transformers do not use a single attention mechanism. Instead, multiple attention mechanisms are utilized to capture diverse perspectives of the input features. Each head (mechanism) learns distinct relationships, improving model robustness and performance.

4. **Feedforward Networks**: Each Transformer layer contains a fully connected feedforward network that applies non-linear transformations to the attended information. Typically, these networks contain two linear layers that have either ReLU or GELU acting as the only activation function between them.

5. **Layer Normalization**: To stabilize training and improve convergence, layer normalization is applied to intermediate outputs within each Transformer layer. This helps prevent exploding or vanishing gradients, ensuring efficient gradient propagation.

6. **Residual Connections**: Transformers use residual (skip) connections to allow direct information flow between layers. This technique mitigates the problem of vanishing gradients in deep networks and improves training stability.

Transformers have been adapted for vision tasks through architectures such as Vision Transformers (ViTs) and Swin Transformers, which replace convolutional layers with self-attention mechanisms. These models have already proven themselves by showing exceptional performance in various computer vision tasks and image generation by effectively capturing global dependencies in visual data.

The Hybrid (CNN+Transformer) Model efficiently processes sequential data with the combination of Convolutional Neural Network (CNN) and Transformers. Here, CNN component is responsible for extracting local, fine-grained features by applying convolutional filters and pooling operations. These operations decrease the spatial dimensions of the feature maps that were extracted from the input data as stated in the research paper [18] by Nukala et al. 2024. Pooling layers also make sure to retain the most significant spatial features while decreasing dimensionality.
The extracted feature representation is then passed to the Transformer component, which captures the long-range dependencies and global contextual relationships within the sequence with the help of self-attention mechanism and multi-head attention mechanism. Layer normalization stabilizes training by normalizing activations, while global average pooling compresses the sequence into a compact global representation, reducing computational overhead.
Finally, the aggregated features are fed into fully connected layers, where non-linear transformations and a softmax activation function produce class probabilities. This hybrid architecture is particularly effective for tasks requiring both local feature extraction and global context understanding, such as time-series classification, signal processing, and bioinformatics. Figure 3.5 provides a simplified flowchart illustrating the workflow of the hybrid model for better comprehension.

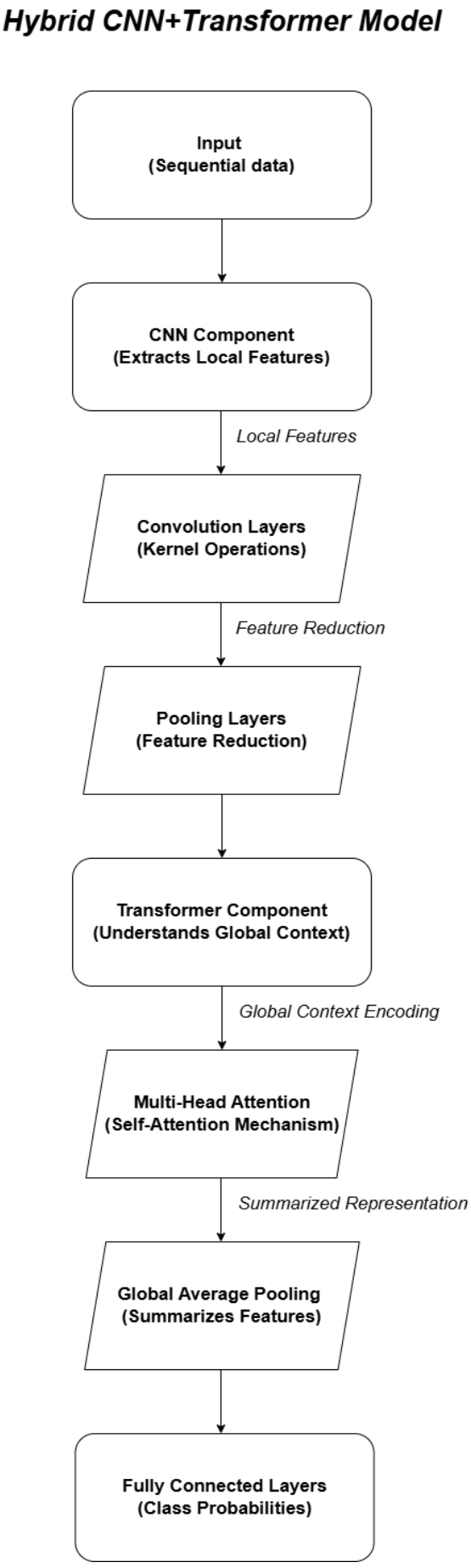

Figure 3.5: The entire workflow of the Hybrid Model

Chapter 4

Model Description

4.1 Model Training and Analysis

We selected our hybrid model based on the fact that this model has not been applied on this area of study. We hoped that this combination can produce better results than other models. Keeping that in mind, we wrote a training code that can incorporate both CNN and Transformers on the extracted features. As we mentioned earlier, hand landmarks and relative position of the hands were saved in a pickle file. This file was used to train our model. To begin, we imported the required Python Libraries including numpy, pickle, tensorflow, Model, Adam, matplotlib.pyplot, sklearn and some others. The remaining workflow can be divided into several segments and the first one is:

Data Preprocessing: In this step, data is loaded from a pickle file which contains features (X), labels (y), and class names (class names). As mentioned in the book [15] by Bhatti et al. 2024, the features (X) are normalized by dividing them by the maximum value, ensuring that the input data is scaled appropriately for the neural network. The dataset is divided into training sets and validation sets (80-20 split) with the help of a Python library named sklearn.

4.1.1 Model Architecture

The proposed hybrid **CNN+Transformers** model is designed to effectively recognize Bengali Sign Language gestures by leveraging both convolutional feature extraction and self-attention mechanisms. The main components of the model's architecture are shown in Figure 4.1. The architecture consists of the following key components:

Input Layer

The model takes as input a sequence of preprocessed feature vectors extracted from Bengali Sign Language videos. Each input sequence is represented as:

$$X \in \mathbb{R}^{T \times F}$$

where T represents the sequence length (number of time steps), and F is the number of extracted features per time step.

The input shape for the model is:

$$(\text{sequence\_length}, \text{num\_features})$$

which ensures that the model can process sequential data effectively.

CNN-Based Feature Extraction

To extract spatial and local temporal features from the input sequence, the model utilizes 1D Convolutional Neural Networks (Conv1D). CNNs are well-suited for capturing patterns within local regions of the sequence, such as hand movements or shape variations over short time spans. The feature extraction process involves the following layers:

- **First Convolutional Layer (Conv1D)**
  - 64 filters
  - Kernel size = 3
  - Activation = ReLU
  - Padding = 'same' (ensures output size remains unchanged)
  - Captures low-level spatial and temporal patterns.
- **Max-Pooling Layer (MaxPooling1D)**
  - Pool size = 2
  - Reduces sequence length by half, retaining only important features while reducing computational complexity.
- **Second Convolutional Layer (Conv1D)**
  - 128 filters
  - Kernel size = 3
  - Activation = ReLU
  - Padding = 'same'
  - Captures deeper hierarchical features, detecting more complex movement patterns.
- **Max-Pooling Layer (MaxPooling1D)**
  - Pool size = 2
  - Further reduces dimensionality, allowing the model to focus on the most salient features.
- **Dropout Layer**
  - Dropout rate = 0.3
  - Prevents overfitting by randomly dropping some neurons during training, making the model more robust.

At this stage, the input sequence has been transformed into a set of high-level feature representations that serve as inputs to the Transformer-based attention mechanism.

Transformer-Based Attention Mechanism

While CNNs are effective for feature extraction, they struggle with capturing long-range dependencies in sequential data. To address this limitation, the model integrates a self-attention mechanism inspired by Transformers. This module allows the model to focus on important time steps within the sequence, ensuring that long-term dependencies are effectively modeled. The attention mechanism consists of the following layers:

- **Layer Normalization**
  - Helps stabilize activations and gradients by normalizing feature distributions.
  - Ensures faster and more stable training.
- **Multi-Head Self-Attention (MHSA)**
  - Number of attention heads = 4
  - Key dimension = 128
  - Each attention head learns a unique representation of temporal relationships within the feature sequence.
  - Computes scaled dot-product attention to focus on the most relevant time steps in the sequence.

  According to the research paper [13] by Vaswani et al. 2023, the self-attention mechanism is formally defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

  where:

  - Q (Query), K (Key), and V (Value) are projections of the input sequence.
  - d_k is the dimension of the key vectors, used for scaling.
- **Global Average Pooling (GAP)**
  - Computes the average of the attention outputs across time steps.
  - Reduces the dimensionality of the sequence while retaining crucial information.
  - Prevents overfitting by replacing fully connected layers with a parameter-free operation.

By leveraging Multi-Head Self-Attention, the model learns to highlight the most significant temporal features of a given sign, improving classification performance.

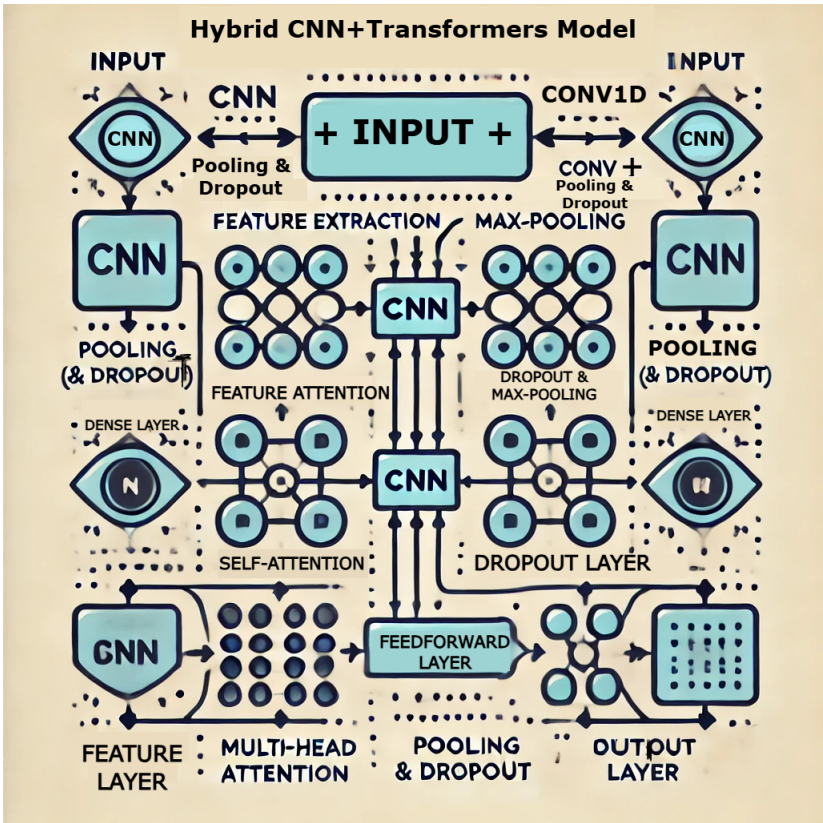

Figure 4.1: Detailed Architecture of the Hybrid Model

Fully Connected Layers (Feedforward Network)

After extracting both **local (CNN) and long-range (Transformers) features**, the processed features are passed through fully connected layers for classification. The final layers include:

- **Dense Layer (Fully Connected Layer)**
  - 128 neurons
  - Activation = ReLU
  - Performs high-level feature abstraction, mapping extracted features to meaningful representations.
- **Dropout Layer**
  - Dropout rate = 0.3
  - Prevents overfitting by randomly deactivating neurons, enhancing model generalization.
- **Output Layer**
  - Softmax activation
  - Number of neurons = num_classes (62) (corresponding to the number of sign language gestures being classified)
  - Converts final feature representations into class probabilities.

Summary of the Architecture

Component	Layers & Operations
Input Layer	Preprocessed sequence data ($T \times F$)
CNN Feature Extraction	Conv1D (64 filters, kernel size=3, ReLU, same padding) → MaxPooling1D (pool size=2) → Conv1D (128 filters, kernel size=3, ReLU, same padding) → MaxPooling1D (pool size=2) → Dropout (0.3)
Transformer Attention Mechanism	Layer Normalization → Multi-Head Self-Attention (4 heads, 128-d) → Global Average Pooling
Fully Connected Layers	Dense (128 neurons, ReLU) → Dropout (0.3) → Dense (num_classes (62), softmax)

Table 4.1: Summary of the CNN+Transformers Model Architecture

This **hybrid CNN+Transformers** model effectively combines spatial feature extraction from CNNs and temporal sequence modeling from Transformers, resulting in a robust architecture for Bengali Sign Language recognition.

4.1.2 Model Training and Evaluation

- **Compilation and Training:**
  - The model is appropriate for multi-class classification applications because it is compiled with the Adam optimizer and sparse categorical cross-entropy loss.
  - To calculate precision value, recall value, and F1-score following every training epoch, a custom callback called MetricsCallback is defined. This aids in evaluating the model's performance at every step of the training.
  - The model is trained for 100 epochs and the training process includes monitoring validation accuracy. The batch size was 32 and the learning rate was kept at 0.0001. We experimented with different learning rates and observed that 0.0001 is the best learning rate for our model. We also tried to vary the number of epochs, however, 100 epochs gave us the highest accuracy.
- **Model Evaluation:** After training, the model is saved as an H5 file (bdslicnntransformers.h5). Here, BDSLI represents the words Bengali Sign Language Interpreter. Then, various plots such as training accuracy, validation accuracy, precision value, recall value, and F1 score were generated after the training was complete. After training the model, we analyzed the accuracies and realized that the model was performing better than we anticipated. Figure

4.2 shows us the training accuracy throughout 100 epochs where the lowest accuracy (8.87%) was seen in epoch 1 and the highest accuracy (99.58%) was recorded for the final epoch with only 1.27% loss. It is also evident from the

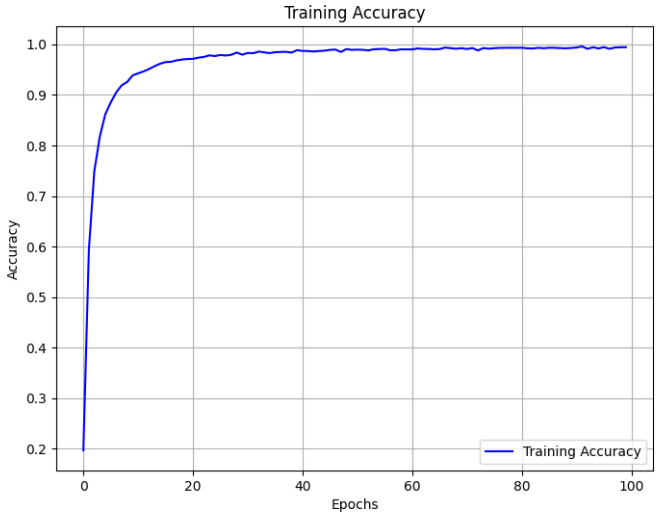

Figure 4.2: Training accuracy of the Hybrid Model

graph that the accuracy rapidly reached 95% within the 13th epoch. After that, the improvement was notably slower. The improvement in validation ac-

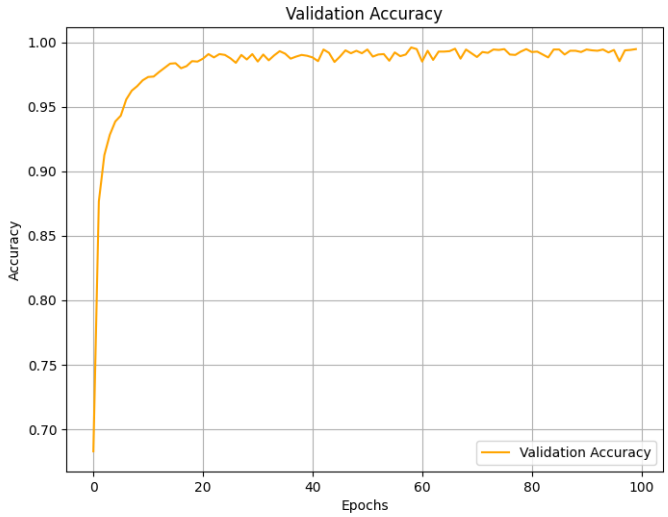

Figure 4.3: Validation accuracy of the Hybrid Model

curacy was somewhat different, as shown in Figure 4.3. The lowest validation accuracy was 68.29% in the first epoch and the final epoch showed 99.48% validation accuracy with 2.07% validation loss. However, the highest values of training accuracy and validation accuracy were almost the same. This indicates that the model was perfectly converging during training. After reviewing both training and validation accuracy, we were certain that our model

had minimal overfitting and achieving a 99.58% training accuracy and 99.48% validation accuracy indicated that our model was performing extremely well on both the training data and the validation data. It will be more transparent if we look at Figure 4.4 where training and validation accuracy are plotted together. This graph makes it apparent that during last few epochs, the validation and training accuracy were close to each other.

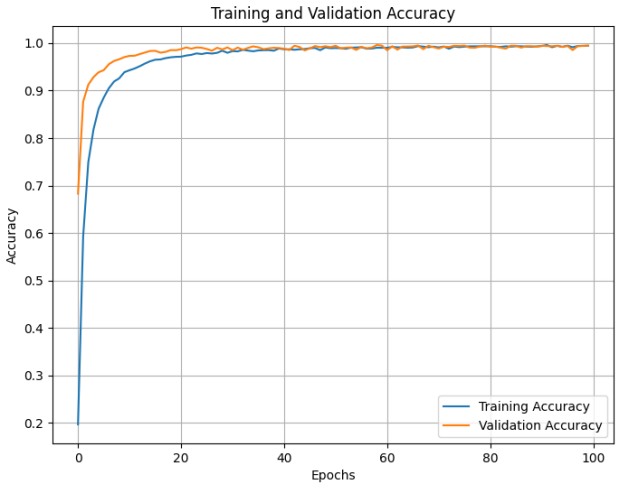

Figure 4.4: Training and Validation accuracy of the Hybrid Model

Nevertheless, we still needed further evidence that our model was indeed performing well. So, we decided to print precision score, recall score, and F1 score after every epoch. Figure 4.5 depicts the gradual improvement in precision value, recall value, and F1 score throughout 100 epochs. In the first epoch, precision was only 70.46, recall was 68.29, F1 score was 64.80. The highest scores were obtained on the final epoch where precision was 99.50, recall was 99.48 and F1 score was 99.48. From these scores, it is apparent that the model has excellent and balanced performance and has minimal misclassification.

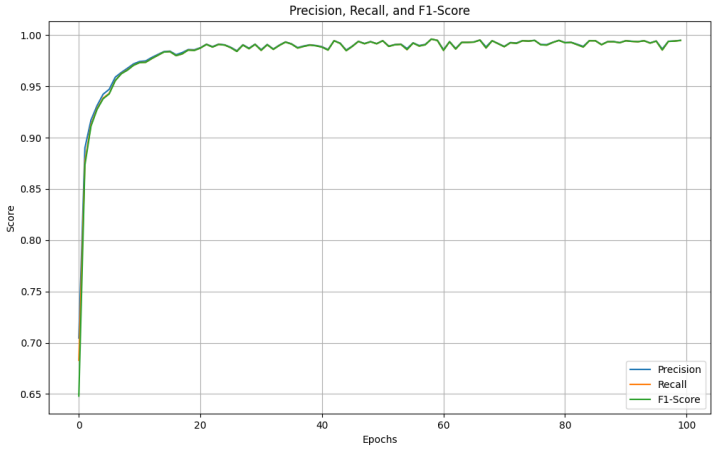

Figure 4.5: Precision, Recall and F1 score of the Proposed Hybrid Model

Lastly, we were surprised to discover that the model completed training in a short time. It only took the model around 19 minutes and 57 seconds to complete 100 epochs. Compared to the other models we trained for this study, this training time was impressive. In the following section, we will compare this training time with other models.

4.2 Evaluating Hybrid Model on Test Dataset

Based on the previous analysis, we were certain that our model was correctly trained and producing exceptional outcomes. Before deploying the model in a web application, we wanted to assess its performance on a test dataset. As mentioned before, we built the test dataset independently from the training dataset to prevent our model from seeing the videos. First of all, we extracted the features (hand landmarks and relative position) from the test dataset, labeled them accurately and saved them in a pickle file. In the next step, we used this pickle file to test our hybrid model by showing the saved features to our model and asking it to determine which classes the features belonged to.

Word	Pre	rec	F1	sup
আজ	0.91	0.98	0.94	50
আবার	0.98	1.00	0.99	50
আমাদের	1.00	1.00	1.00	50
আমার	1.00	1.00	1.00	50
আমি	1.00	1.00	1.00	50
আসা	0.94	1.00	0.97	50
কবে	1.00	1.00	1.00	50
কাজ	1.00	1.00	1.00	50
কিভাবে	0.94	0.94	0.94	50
কী	1.00	1.00	1.00	50
কে	1.00	1.00	1.00	50
কেন	1.00	1.00	1.00	50
কোথায়	1.00	0.96	0.98	50
কয়টা বাজে	0.98	1.00	0.99	50
খাওয়া	1.00	1.00	1.00	50
ছেলে	1.00	0.86	0.92	50
ডাকা	1.00	1.00	1.00	50
তাদের	1.00	1.00	1.00	50
তুমি	1.00	0.98	0.99	50
তোমাদের	0.96	0.94	0.95	50
দুঃখিত	1.00	1.00	1.00	50
দুপুর	1.00	1.00	1.00	50
দেখতে	0.94	1.00	0.97	50
দেখা	1.00	1.00	1.00	50
দেখা হয়ে	1.00	0.96	0.98	50
দেশ	1.00	0.92	0.96	50
দয়া করে	1.00	1.00	1.00	50

Word	Pre	rec	F1	sup
ধন্যবাদ	1.00	1.00	1.00	50
না	1.00	1.00	1.00	50
নাম	1.00	1.00	1.00	50
পড়া	0.98	1.00	0.99	50
বন্ধু	0.96	0.98	0.97	50
বলা	1.00	1.00	1.00	50
বাংলা	0.98	1.00	0.99	50
বাবা	1.00	1.00	1.00	50
বাড়ী	1.00	1.00	1.00	50
বুধবার	1.00	1.00	1.00	50
বৃহস্পতিবার	1.00	0.98	0.99	50
বোন	1.00	1.00	1.00	50
ভাই	1.00	1.00	1.00	50
ভালো	0.98	1.00	0.99	50
ভাষা	0.98	1.00	0.99	50
মঙ্গলবার	0.86	1.00	0.93	50
মা	0.96	0.96	0.96	50
মেয়ে	0.98	1.00	0.99	50
যাওয়া	1.00	0.92	0.96	50

Word	Pre	rec	F1	sup
রবিবার	0.98	1.00	0.99	50
রাত	0.98	0.98	0.98	50
লেখা	1.00	0.98	0.99	50
শনিবার	0.96	1.00	0.98	50
শুক্রবার	1.00	0.98	0.99	50
শোয়া	0.98	1.00	0.99	50
সকাল	1.00	1.00	1.00	50
সাহায্য	0.96	1.00	0.98	50
সিনেমা	1.00	1.00	1.00	50
সুন্দর	1.00	1.00	1.00	50
সে	1.00	1.00	1.00	50
সোমবার	1.00	0.90	0.95	50
স্ত্রী	1.00	0.98	0.99	50
স্বামী	1.00	0.98	0.99	50
হবে	1.00	0.98	0.99	50
হ্যা	1.00	1.00	1.00	50
acc			0.99	3100
m.avg	0.99	0.99	0.99	3100
w.avg	0.99	0.99	0.99	3100

Figure 4.6: Model's performance on the test dataset

Following that, we compared the predicted class labels with the actual labels and calculated the accuracy, recall and precision for each Bengali word. Figure 4.6 demonstrates how accurate our model was for each word. The overall accuracy of our model was 98.65.% for 3100 videos and both the micro average and weighted average were 99.00%. Most of the Bengali words were flawlessly predicted by our model with a small number of exceptions. Words including 'Mongonbar', 'Dekhte', 'Kivabe', 'Maa', 'Bondhu', 'Tomader', 'Desh', and 'Shombar' were occasionally misclassified. We anticipated that our model will have problems with 'Mongolbar' and 'Maa' because these two words have practically the same hand gestures.
The issue with the signs for "Kivabe" and "Tomader" was that the signs prevented me from clearly displaying my hands to the camera. Hence, the hand landmarks were not correctly extracted, leading to inaccurate predictions. The hand sign for the word 'Bondhu' required using both of my hands where one hand was covering another. Sometimes, the model could not detect both hands and was giving wrong predictions depending on the only hand visible to the model. Lastly, the words 'Desh' and 'Shombar' were sometimes guessed incorrectly by the model because of the similarity these words have with 'Dhonnobad' and 'Valo' respectively.
A confusion matrix presented in Figure 4.7 was plotted showing the accuracy of the predictions our model made on the test dataset. For a multi-class classification

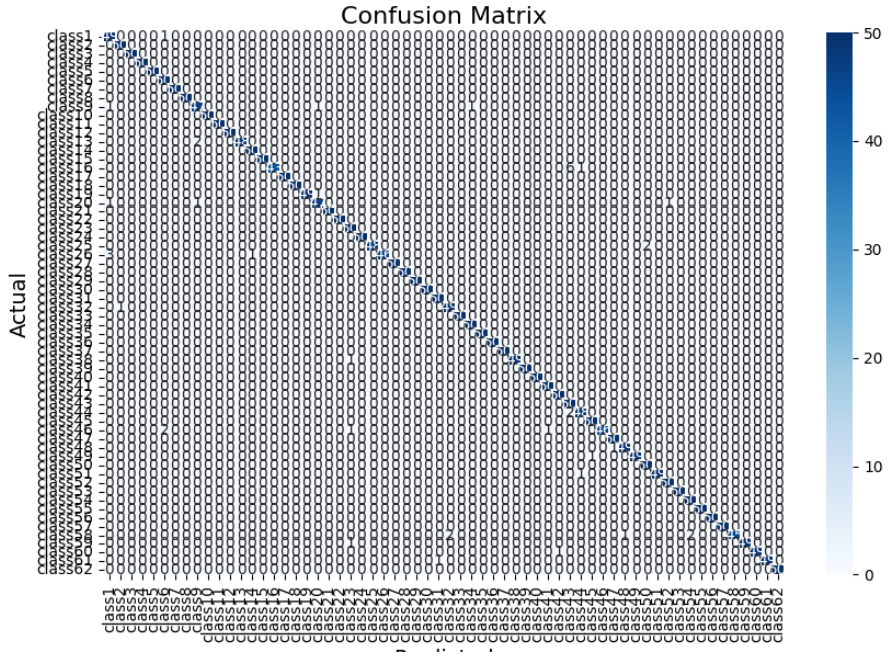

Figure 4.7: Confusion Matrix depicting the performance of the model on test dataset

problem, the confusion matrix is an $N \times N$ matrix, where N is the number of classes as mentioned in the research paper [10] by Ghayoumi (2023). The general equation for each element of the confusion matrix is:

$$C_{ij} = \sum_{n=1}^{N} \mathbb{1}(y_n = i, \hat{y}_n = j)$$

where according to an article [23] found from arxiv.org:
- C_{ij} is the number of times a true class i was predicted as class j.
- y_n is the true label of the n-th sample.
- $\hat{y}_n$ is the predicted label of the n-th sample.
- $\mathbb{1}(\cdot)$ is an indicator function that equals 1 if the condition inside holds and 0 otherwise.
- N represents the amount of samples present here.

Here, the names of the word classes on the Y-axis stand for the actual classes and the words shown on the X-axis are what the model recognized. The diagonal blue line represents the extraordinary performance displayed by our Hybrid CNN+Transformers model. To be more precise, the diagonal line tells us that the model was accurately labeling the word classes with infrequent inconsistency. Instead of showing the class names in Bengali, we used generic names like class1, class2 etc. on the confusion matrix. Here, class1 represents the word 'Aj' and class 62 represents the words 'Ha' which means the classes are organized in alphabetical order.

Chapter 5

Experimentation and Result Analysis

5.1 Model Training and Comparison

We have already discussed the training process of our CNN+Transformers model and tested it using a test dataset. Now, it was time for us to see if our nominated model was indeed the best performing one for Bengali Sign Language Recognition. As mentioned earlier, we chose models that have already been used by other researchers for sign language interpretation. From all the models we came across while going through previous research papers, we selected some of the best performing models namely CNN+LSTM, GRU, LSTM, TCN, Transformers, CNN+BiLSTM and CNN+GRU. High-performing models like YOLOV8 and Mobilenetv2 were ignored, since they work better with image datasets than with video datasets. At first, we worked with YOLOV8 but realized that even though this model can only do well for Bengali sign words that are simple and do not include any hand movements. However, we have numerous words that have complicated hand movements. Hence, we dropped our idea of using YOLOV8 and moved to more complex models that can handle elaborate hand gestures.
We trained all of our models with the same pickle file we used to train our primary CNN+Transformers hybrid model. After training the 7 aforementioned models, we tested the models together on the test dataset we prepared. In the next part of this section, we will provide the details on training 7 models and compare the accuracies of the models on the test data.

5.1.1 Hybrid (CNN+LSTM) Model

This model is widely used in sign language translation and recognition. The research paper [5] written by Elhagry et al. 2021 also uses this model. So, it was an obvious choice for us to compare our model with this state-of-the-art model. At first, we imported the necessary Python Libraries and loaded our pickle file. The architecture of this model is similar to that of our model. This model employs the Functional API design, starting with an input layer shaped for sequential time-series data. The CNN component includes two Conv1D layers (64 and 128 filters, kernel size of 3, ReLU activation) to capture local patterns and MaxPooling1D layers to down-sample significant features. The LSTM component, with 128 units and tanh activation, learns

long-term dependencies and passes the final output only (return-sequences=False) to the next stage. Finally, fully connected layers with 128 ReLU-activated units and dropout prevent overfitting, while a softmax layer outputs class probabilities. We kept the learning rate at 0.0001 and trained the model for 100 epochs. We iterated the training process multiple times and figured out that this is the best setup to train this model. The batch size was 32 and this model was also saved in a H5 file (bdsli-cnnlstm.h5). It took the model around 17 minutes and 47 seconds to complete 100 epochs and after the model was trained, different graphs were plotted for training accuracy, validation accuracy, precision score, recall value, and F1 score. Figure 5.1 shows the training and validation accuracy of the model. If we closely

Figure 5.1: Training accuracy, Training loss, Validation accuracy and Validation loss for CNN+LSTM Model

observe Figure 5.1, we will see that the model had the lowest accuracy in the first epoch as expected and the highest training accuracy was achieved in the last epoch. The highest training accuracy was 99.37% with a loss of 2.21% which is close to our CNN+Transformers model. In addition, this model had 98.65% validation accuracy with 3.90% validation loss. The precision value, recall value, and F1 score for this trained architecture were 98.67, 98.65 and 98.64 sequentially. These values prove that our model had better training and validation accuracy than CNN+LSTM. However, compared to CNN+LSTM, our model required two extra minutes to complete the training procedure. We needed to use the test dataset to determine whether the additional time was worth it or not.

5.1.2 Gated Recurrent Units (GRU) Model

We have already mentioned that all of the models we trained for this study were trained on the same pickle file and GRU model was not any different. In the initial step, we imported necessary Python libraries, loaded the pickle file and initialized the model. The model is a GRU-based neural network designed for sequence classification and this architecture applies two different GRU layers. One of the layers has 64 units and the other has 128. These two layers can process sequential data

by capturing temporal dependencies. Then a dense layer is utilized which has the responsibility of extracting features. Another layer called dropout layer is used to prevent the model from overfitting on the training data. After that, the final output layer which contains softmax activation function gives the class probabilities as output. The model uses sparse categorical cross-entropy as the loss function and the Adam optimizer to adjust weights during training. In this way, the model learns to map input sequences to their corresponding class labels effectively.
This model was trained for 100 epochs with a batch size of 32. The learning rate was 0.0001 just like the previous two models we have trained. After the training was complete, the model was saved in an H5 file (bdsli-grumodel.h5). Several graphs were plotted to demonstrate the training and validation accuracy. In addition, other performance metrices such as precision value, recall value, and F1 score were also printed after each epoch to showcase the performance of the trained model. Figure

Figure 5.2: Training accuracy, Training loss, Validation accuracy and Validation loss for GRU Model

5.2 illustrates the training and validation accuracy together and from this graph it can be seen that the first epoch had the lowest training and validation accuracy as usual. The model achieved the highest training accuracy (97.88%) in the last epoch with a loss of 6.81%. The validation accuracy recorded in the last epoch was 95.94% with 14.70% validation loss. Moreover, the final precision value, recall value, and F1 score of this model were 96.13, 95.94 and 95.92 respectively. Lastly, the total time taken by the model to complete 100 epochs was 47 minutes and 39 seconds. The overall performance was good enough to assume that this model will work well for sign language recognition task. However, from the provided data, we can deduce that GRU model is inferior to CNN+Transformers and CNN+LSTM for our dataset. Still, we wanted to test GRU model on our test dataset before jumping to any conclusions.

5.1.3 Long Short Term Memory (LSTM) Model

After training the GRU model, it was time to try out another model. The next model we have chosen for this study is LSTM model which is short for Long Short Term Memory. For training this model, we had to import some required Python libraries and load our saved pickle file. LSTM model is one kind of neural network which was designed for sequence classification tasks. It processes sequential input data using two stacked LSTM (Long Short-Term Memory) layers. The first LSTM layer (with 64 units) outputs sequences and allows the second LSTM layer (with 128 units) to capture long-term dependencies in the data even further. LSTM model is effective when we are dealing with time-series data. LSTM can handle this type of time-series data by learning temporal patterns from the data and by using its gating mechanisms to manage the vanishing gradient issues. The outputs from the second LSTM layer are passed to a dense layer with 128 units and ReLU activation. This dense layer extracts the high-level features. A dropout layer with a 30% rate has the ability to disable neurons randomly during the training process which can help prevent overfitting. Then the output layer which uses softmax activation produces class probabilities. These probabilities are used to predict the label of a given input data. This LSTM model was trained with Adam optimizer and sparse cross-entropy as the loss function. These are used owing to the fact that both of these work well with classification tasks where multiple classes are present and the classes are labeled as integers.

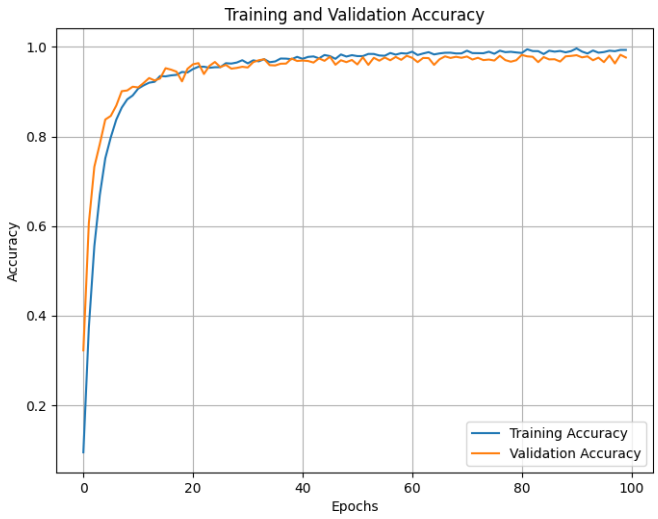

Figure 5.3: Training accuracy, Training loss, Validation accuracy and Validation loss for LSTM Model

The hybrid CNN+LSTM model uses CNN for extracting spatial features from frames which enhances the ability to capture patterns from images and videos. LSTM is used to process the temporal dependencies. The combination of CNN and LSTM has the ability to capture both spatial and temporal dependencies. As a result, the recognition accuracy is increased compared to the LSTM model which is used without CNN. When we use LSTM model only, it solely focuses on temporal dependencies and ignores spatial features. Technically, an LSTM model is supposed

to have worse performance than a hybrid CNN+LSTM model. From Figure 5.3, we can see that the highest accuracy (99.36%) was reached in the final epoch with a loss of 2.19%. These values are almost the same as the CNN+LSTM model. The highest validation achieved by the LSTM model was 97.61% with a loss of 8.10% which is comparatively lower than CNN+LSTM. Hence, we can say that a hybrid CNN+LSTM model will perform better than LSTM on unseen data. The precision value, recall value, and F1 score for the trained LSTM model were 97.74, 97.61 and 97.61 sequentially. Moreover, it took more than 37 minutes for the LSTM model to complete 100 epochs, whereas the CNN+LSTM model only took about 18 minutes. Nevertheless, we wanted to demonstrate using our test dataset that CNN+LSTM is superior than LSTM and that the CNN+Transformers model we selected performs better than any other model.

5.1.4 Temporal Convolutional Network (TCN) Model

The Temporal Convolutional Network (TCN) architecture uses a number of convolution layers to process sequential data. However, before using these layers, this architecture increases the dilation rates in order to effectively capture temporal dependencies. Then the TCN block combines causal convolution, batch normalization, dropout and residual connections. Here, the model uses dropout for regularization, whereas the residual connections are used for ensuring a stable training process. The model stacks multiple TCN blocks, performs global average pooling, and applies dense layers to produce predictions across classes using a softmax activation function. This approach uses temporal relationships in the data to classify hand gestures effectively. We also trained this model for 100 epochs with a learning rate of 0.0001. After training, multiple graphs were plotted to show the training accuracy, validation accuracy, precision, recall and F1 scores for the trained model. Figure 5.4 illustrates the training and validation accuracy together in a line graph. From Figure 5.4, it is evident that the highest training accuracy reached by the

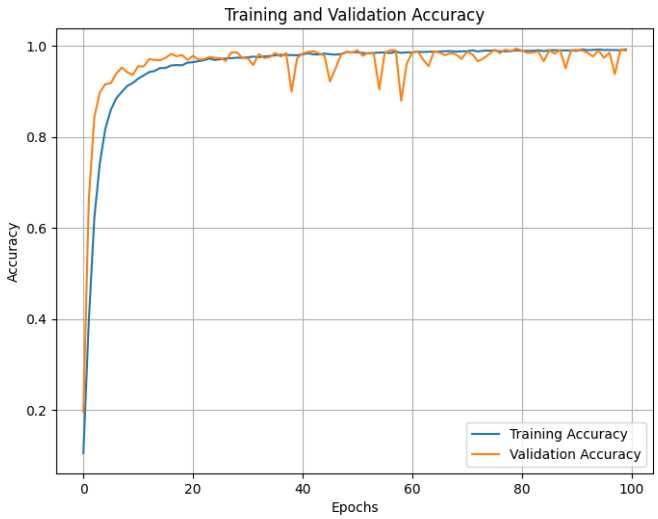

Figure 5.4: Training accuracy, Training loss, Validation accuracy and Validation loss for TCN Model

TCN model in the last epoch was 99.29% with 2.23% training loss. The highest validation accuracy recorded for this model was 98.94% where the validation loss was 4.13%. There were comparatively more fluctuations in the validation accuracy than in the previous models. However, these fluctuations are typical and might result from random variations in the validation set, and they do not indicate any significant issue. The precision value, recall value, and F1 score were 99.03, 98.94 and 98.95 consecutively. Around 28 minutes and 15 seconds was spent by the model to finish 100 epochs. We will also examine how well this model can perform on our test dataset and compare the results with other models.

5.1.5 Transformer Models

This standalone Transformer architecture utilizes self-attention mechanism in order to capture complex relationships in sequential data. The transformer block also incorporates multi-head attention so that it can focus on different parts of the input data simultaneously. The multi-head attention is followed by feed-forward layers which are responsible for processing the data further. This transformer block also uses residual connections and layer normalization in order to ensure stable gradient flow and effective learning. Multiple transformer blocks like this are stacked together in order to improve the ability of the model to capture temporal dependencies. Following the transformer layers, there is another layer called global average pooling layer. This particular layer converts the input sequence into a representation of a certain size. This particular representation then goes through some dense layers and an output layer with softmax activation function. This design enables the model to efficiently learn temporal and contextual patterns, making it well-suited for tasks involving sequential gesture recognition.

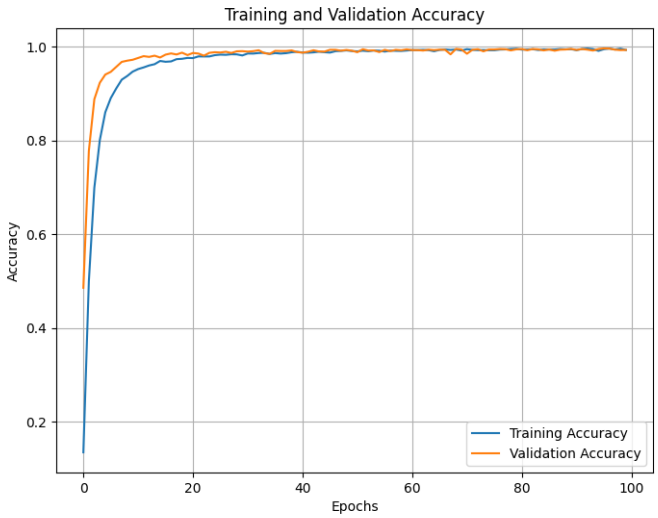

Figure 5.5: Training accuracy, Training loss, Validation accuracy and Validation loss for Transformer Models

We had high expectations for this architecture because it is frequently used for NLP tasks. The model was trained for 100 epochs with a learning rate of 0.0001 just like previous 4 models. When the training was complete, we were surprised to see

the performance. Figure 5.5 depicts the overall performance of this model during training.
Even though transformers yielded better results than most of the models we have trained so far, it was still inferior to the hybrid model (CNN+Transformers) we chose. The reason behind it is that Transformers rely entirely on self-attention to model both local and global relationships, which can be computationally intensive and less efficient at learning fine-grained local features. On the contrary, the hybrid (CNN+Transformers) model uses CNN for extracting local features. It uses Transformers to capture long-range dependencies and global context. It makes the combination particularly effective for tasks like sign language recognition.
If we examine Figure 5.5, we will see that transformer model's peak training accuracy was 99.34% with a loss of only 2.08%. This architecture achieved the second highest validation accuracy (99.35%) among all the models we worked with in this study with 3.69% validation loss. On the other hand, this model took the highest training time (53 minutes and 36 seconds) which is significantly higher than CNN+LSTM and CNN+Transformers. Lastly, the precision value, recall value, and F1 score for this model were 99.41, 99.35 and 99.36 respectively. In the following subsection, the performance of this model will be compared to that of other models using our test dataset.

5.1.6 Hybrid (CNN+BiLSTM) Model

The two components of this hybrid model have different functionalities. First of all, the CNN component utilizes two convolutional layers. These layers have ReLU activation function and max pooling operations in them so that the CNN can learn spatial patterns from the input data such as images and videos. A dropout layer is used to prevent the model from overfitting. After that, the Bidirectional LSTM layers help the model to capture temporal dependencies. Previously in the CNN+LSTM model, the LSTM layers captured the temporal dependencies one dimensionally. However, the BiLSTM layers capture them from both directions of the input sequence. In the following step, the fully connected layers improve the features which were extracted from the input data. Lastly, the final layer uses softmax activation function to predict the class probabilities and show them as output.
In the research paper [11], Krishnaiah et al. 2023 used a similar version of this hybrid architecture. Another study [3] conducted by Abdul et al. 2021 also used another version of this model. This model also uses sparse categorical cross-entropy as the loss function and Adam optimizer with a low learning rate for a stable training process. Just like our previous models, this hybrid CNN+BiLSTM model was trained for 100 epochs with a batch size of 32 and a learning rate of 0.0001.
Following the completion of the training, different line graphs were plotted to illustrate training and validation accuracy, precision value, recall value, and F1 score. Figure 5.6 demonstrates the overall training and validation accuracy of this model. If we look at Figure 5.6 closely, it will be clear that this model achieved exceptional training and validation accuracy. The highest training accuracy recorded was 99.32% with the lowest training loss of 2.19%. The highest validation accuracy (99.00%) was also reached in the final epoch with 3.78% validation loss. For this architecture, the precision value, recall value and F1 score were 99.04, 99.00 and 99.00 respectively. This model took around 19 minutes to complete the training process.

Figure 5.6: Training accuracy, Training loss, Validation accuracy and Validation loss for CNN+BiLSTM Model

From these values, it is evident that this CNN+BiLSTM model will outperform the CNN+LSTM in all the criteria except the total training time.

5.1.7 Hybrid (CNN+GRU) Model

We have already trained a GRU model for this study and it was now time to try out a hybrid version of GRU and compare it with other models. This CNN+GRU model is a hybrid architecture that combines Convolutional Neural Network (CNN) and Gated Recurrent Units (GRU) to process sequential data. In this hybrid architecture, CNN has the responsibility to extract features from the input sequence. CNN has two 1D convolutional layers that have ReLU activation function and max-pooling operations in them. These two layers decrease the spatial dimensions of the input sequence and capture the most important local patterns from the data. Then overfitting is prevented by applying dropout.
When the CNN component is done with extracting features, the GRU layer captures the temporal dependencies in order to process the sequential data. This layer then creates a feature vector from these captured dependencies. After that, the feature vector goes through a fully connected layer which has ReLU activation function in it. Lastly, the vector goes through a final output layer with softmax activation function so that the model can classify the input image or video into a pre-defined class. The model uses the Adam optimizer with a low learning rate of 0.0001 and sparse categorical cross-entropy as the loss function.
The model was trained for 100 epochs with a batch size of 32. After training was complete, several graphs were plotted to show the overall performance of the model. Figure 5.7 depicts the gradual increase in training and validation accuracy with epochs. As GRU model had the worst performance among all the models we trained, we were not expecting much from the hybrid version of the GRU model. However, we were surprised to see that the hybrid CNN+GRU model performed a lot better than GRU. If we closely examine Figure 5.7, we will see that the hybrid

Figure 5.7: Training accuracy, Training loss, Validation accuracy and Validation loss for CNN+GRU model

CNN+GRU reached the highest training accuracy (98.39%) in the 100th epoch with 5.31% training loss. In the same epoch, the validation accuracy was 97.97% with a validation loss of 6.48%. Moreover, the precision value, recall value, and F1 score were 98.06, 97.97 and 97.96 consecutively. Lastly, total training time for this hybrid model was only 12 minutes and 29 seconds which is the lowest training time we have seen in this study. All of these data indicate that the CNN+GRU model is a huge upgrade from the standalone GRU model.

5.2 Analysis and Comparison of Training Data

In the previous section, we have observed the training and validation accuracy obtained after training 7 other models with the training data. The same video dataset was used to train all the models in this study so that we can compare their performance with our CNN+Transformers model. Previously, we saw the training data and graphs for each model separately. Now, we want to compare them together using two tables where the first table will show the training accuracy, validation accuracy and total training time. The second table will show and compare the precision values, recall values and F1 scores that were recorded for the trained models.

From Table 5.1, it is evident that our model achieved the highest training and validation accuracy which are 99.58% and 99.48%. CNN+LSTM obtained the second highest training accuracy (99.37%), whereas the second highest validation accuracy (99.35%) was achieved by Transformers. GRU model was the worst-performing one on this list with a training accuracy of 97.88% and a validation accuracy of only 95.94%. This model took almost 48 minutes to train which is second to the longest time (53m 36s) taken by Transformers. However, when we combined GRU with CNN, the performance improved significantly and this combination took the lowest amount of time (12m 29s) to complete training. Even though CNN+GRU had the lowest training time, the training and validation accuracy were still lower than most

Model	Train acc	Val acc	Training Time
CNN+Transformers	**99.58%**	**99.48%**	19m 56s
CNN+LSTM	99.37%	98.65%	17m 46s
GRU	97.88%	95.94%	47m 39s
LSTM	99.36%	97.61%	37m 4s
TCN	99.29%	98.94%	28m 15s
Transformers	99.34%	99.35%	53m 36s
CNN+BiLSTM	99.32%	99.00%	18m 57s
CNN+GRU	98.39%	97.97%	**12m 29s**

Table 5.1: Training accuracy, Validation accuracy and Training Time with their respective models

models. Our model (CNN+Transformers) took around 20 minutes to finish training which is the fourth lowest training time on the table. Still, the training time was not considerably higher than other models and the extra time was justifiable as the model was outperforming other models.

After analyzing training and validation accuracy, we still wanted to look at the precision values, recall values and F1 scores that were recorded for each model during the training process. Table 5.2 clearly demonstrates these scores and marks the highest values using bold font. If we closely examine this table, it will be apparent that CNN+Transformer has the highest values which are 99.50, 99.48 and 99.48. The model that has the second best scores was Transformers with values 99.41, 99.35 and 99.36. As anticipated, the GRU model has the lowest results, while the LSTM model performs slightly better. Another important thing to note here is that both TCN and CNN+BiLSTM have better values than CNN+LSTM even though CNN+LSTM had higher training accuracy.

Model	Precision	Recall	F1 Score
CNN+Transformers	**99.50**	**99.48**	**99.48**
CNN+LSTM	98.67	98.65	98.64
GRU	96.13	95.94	95.92
LSTM	97.74	97.61	97.61
TCN	99.03	98.94	98.95
Transformers	99.41	99.35	99.36
CNN+BiLSTM	99.04	99.00	99.00
CNN+GRU	98.06	97.97	97.96

Table 5.2: Other Performance Metrics to Compare the models

All of these data depicted in these tables indicate that our model is capable of outperforming other models in predicting different Bengali sign words from given videos. We were hopeful that our model will achieve the highest accuracy on the unseen test dataset. The only model we are concerned about is Transformers even though Transformers took significantly more time to train. We are concerned because if Transformers has better test accuracy than our CNN+Transformers model, we will have to switch to Transformers and make it our primary model. We also

think CNN+BiLSTM and TCN will perform well for our test dataset. The next subsection will cover how each model performed on the test dataset.

5.3 Evaluating Performance on the Test Dataset

As we have mentioned earlier, we prepared a separate test dataset to evaluate all the trained models on unseen videos. At first, we extracted the hand landmarks and the relative position of the hands from the videos and saved them in a pickle file. The features were saved in X and the class labels were saved in y. After that, we loaded all of our trained models and showed the features saved in X to them. The models then predicted the labels from the features and we compared what they predicted with the real class labels. It is clearly demonstrated in Figure 5.8 how successfully each model performed this task. In Figure 5.8, the highest accuracy is

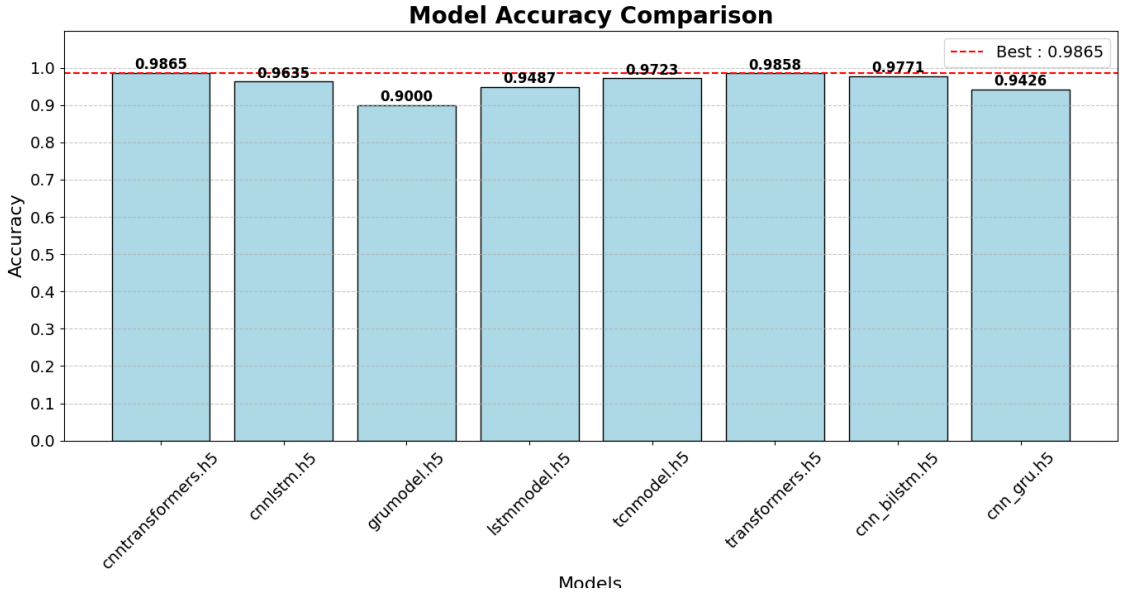

Figure 5.8: Performance displayed by the models on Test Dataset

98.65% which is achieved by our hybrid (CNN+Transformers) model. The second and third highest accuracy (98.58% and 97.71%) are obtained by Transformers and CNN+BiLSTM sequentially.

In contrast, GRU model has the lowest test accuracy which is 90.00%. This is a good accuracy for a object detection model, however, compared to other models here, this value is considerably low. The hybrid CNN+GRU model has the second lowest accuracy on the test data which is 94.26%. These percentages indicate that our primary model outperforms the others, with Transformers being the only model that can closely match its performance.

We also wanted to analyze the precision values, recall values, and F1 scores obtained by each model when they were tested with the test dataset. Figure 5.9 depicts the models and their respective scores in 3 separate histograms. From the first bar graph, it can be noticed that CNN+Transformers has the highest precision score which is 98.71, whereas GRU model gives us the worst precision value (90.76). With a precision score of 98.70, the Transformers-based model takes second place as expected. The precision scores obtained by the remaining models are below 98.

Similarly, the second graph demonstrates that our hybrid model has the highest score (98.65) for recall with Transformers taking the second place with a score of 98.58. Among the other models, only TCN and CNN+BiLSTM have scores higher than 97. Lastly, CNN+Transformers has a F1 score of 98.64 which is the highest value on the chart, while GRU has the lowest value of 89.62. The F1 score for Transformers and CNN+BiLSTM are 98.58 and 97.65 respectively.
Transformers models seem to perform better than other models except for our hybrid CNN+Transformers model. This further indicates that choosing Transformers for this study was a great idea. All of these figures and data prove the fact that we have

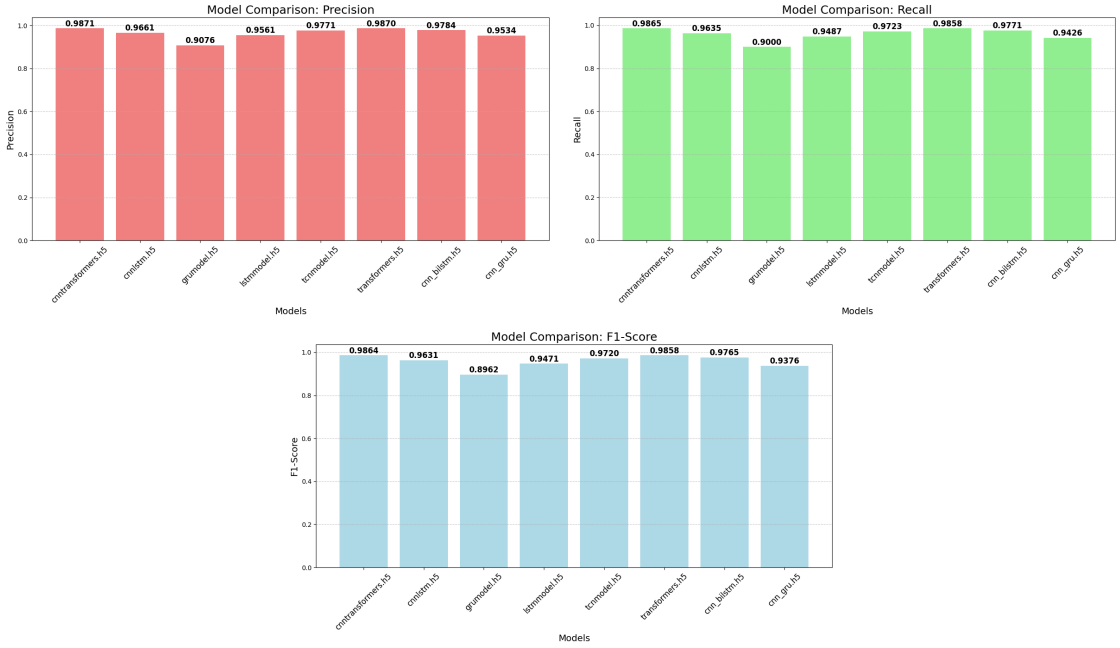

Figure 5.9: 3 Different Performance Metrics to Evaluate the Models on Test data

chosen the best model for this study which can outperform other models on both training and test dataset. Our model has the best values for all the categories we used here to evaluate the models. Now that we have proven our claim, we are ready to deploy our hybrid model into our web app and examine how accurately the app can predict Bengali sign words from the webcam and how efficiently it can produce meaningful sentences with those words.

5.4 Deploying Model and Analyzing Predictions

After showing that we have chosen the best performing model for our study, it was time to deploy the model in a web application. In order to make that happen, we had to design a webpage and write a proper prediction code first. In the prediction code, we introduced some conditions such as a prediction delay of 2 seconds, which means the model will wait 2 seconds before predicting a new word. Another condition was that the model will not predict the same word twice in a row, instead, will wait for a new sign to be shown on the camera. After writing the prediction code and running it, we looked at how accurate our model was in recognizing sign words live from the webcam. We have discovered that the model was working better than we

Figure 5.10: Web App interpreting the sentence "Dhonnobad! Abar Ashben"

anticipated. Seeing that, we incorporated the same prediction code in our app.py file. Then we started the app and started showing the Bengali sign words on the webcam.
Figure 5.10 shows us the words we showed to the camera, and in the last picture, the final output is shown as a full Bengali sentence. The words we have shown in Bengali sign language were "Dhonnobad", "Abar" and "Asha" sequentially. The app was able to not only interpret all of these words, it also constructed a full sentence with proper punctuation marks with the predicted words. The full sentence means "Thank you! Please visit us again" in English. If we look closely, we will see that even though the last word we showed was "Asha", the finalized sentence changed this word to "Ashben" and added some punctuation marks in the right places. This was another condition that we added to our app.py code.

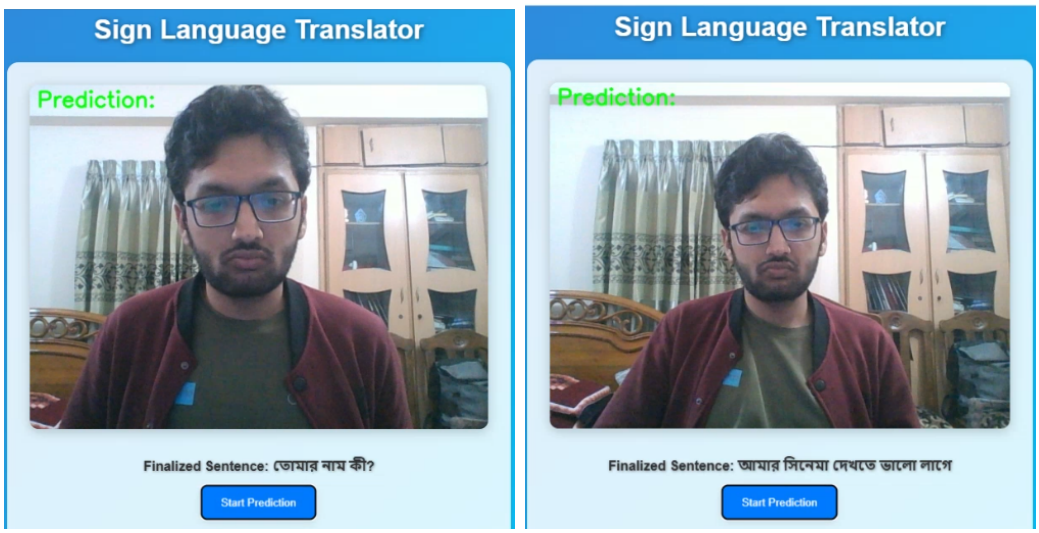

Figure 5.11: Web App showing the sentences "Tomer naam ki?" and "Amar cinema dekhte valo lage"

We were planning to use NLP here to transform sentences when necessary; however, we did not collaborate with any NLP specialist for this paper and wanted to focus on

comparing the models and finding the most suitable one. We can call it a limitation of our study and we will try to work with this limitation in our next research paper. Another sentence we tried to predict was a simple one, which is "Tomer naam Ki?". This sentence translates to "What is your name?". Then we made our app predict a large number of different sentences. We estimated that with the dataset of 62 different words, our model can construct 100+ different sentences accurately. Figure 5.11 shows the sentences "Tomer naam ki" and "Amar cinema dekhte valo lage" which can be translated as "I like watching movies". As the second sentence was too long and the first sentence was too short, we did not show the signs for different words like we showed before.

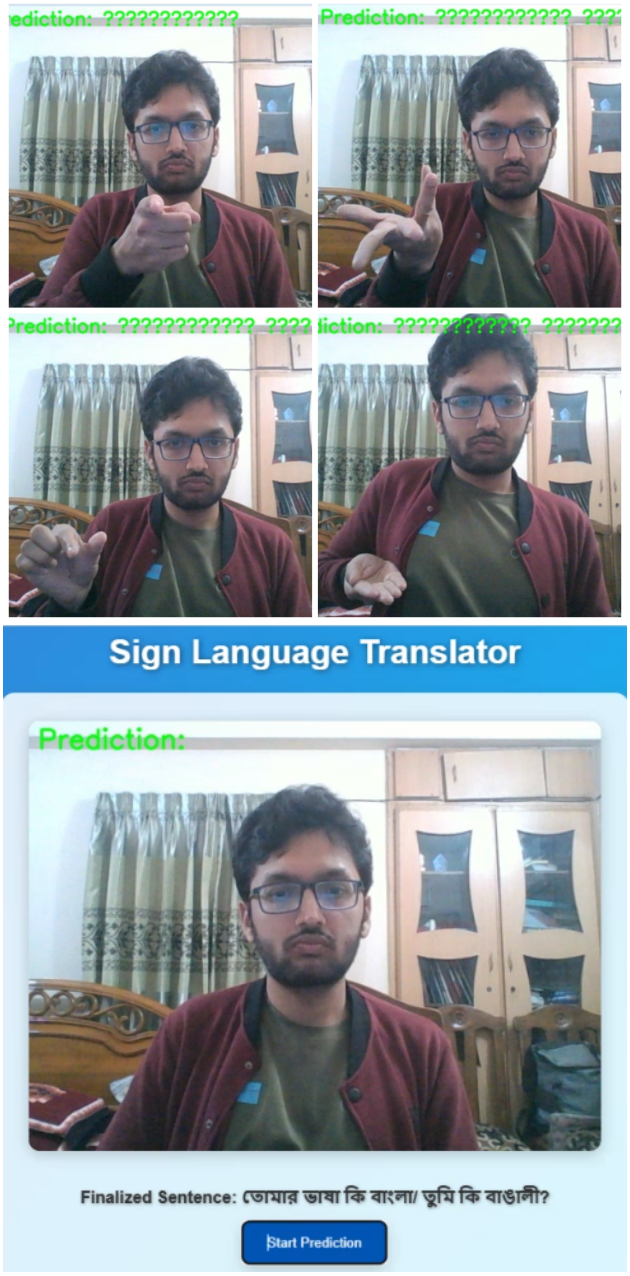

Figure 5.12: Web App predicting the sentence "Tumi ki Bangali?"

The last sentence we want to demonstrate here uses 4 words. The words are "Tumi", "Vasha", "Ki" and "Bangla" which will produce a sentence "Tumi vasha ki Bangla".

However, this sentence does not sound correct. Hence, we had to use the sentence transformation here. We wrote a condition in our code where if the model sees these 4 words in this same sequence, it will slightly change some of the words and then print the sentence with those modified words.
As you can see in Figure 5.12, some of the words in the finalized sentence look different. The sign in the upper left picture in Figure 5.12 represents the word "Tumi". The lower left photo represents the word "Vasha" and the photo in the upper right corner shows the sign for "Ki". Lastly, the lower right picture depicts the sign for the word "Bangla" and the last picture demonstrates the full sentence output. Here, "Tumi" becomes "Tomer" as the signs for both of these words are the same.
The remaining 3 words do not change; however, a question mark is added at the end of the sentence because someone is asking a question here in this sentence. "Tomer vasha ki Bangla?" or "Tumi ki Bangali?" means "Is Bengali your mother tongue?" in English. These are only a handful of the numerous sentences that our web app can interpret if the appropriate sign words are shown on the camera. Most of the words are recognized by the model; however, the model sometimes gets confused with words that have similar hand signs. For example, our model occasionally misclassifies the words "Amar" and "Amader" because both of these words have similar hand gestures. Some other pairs such as "Vasha" and "Shahajjo", "Maa" and "Mongolbar", "Ki" and "Kothay" are also confusing the model. These issues are not severe, as we believe that they can be fixed with a better dataset with better video quality.

5.5 Statistical Analysis with Existing Models

Previously, we have trained seven different models and compared them with our proposed hybrid model. Now, we will show a comparison between our model and the models proposed by other researchers. All of the research papers which will be used here are already mentioned in Chapter 2. Here, the models will be compared using different categories, and it will be proven that our model outperforms all the others for the sign language recognition task. Table 5.3 clearly shows the comparison between various models, and makes it evident that the hybrid (CNN+Transformers) model has the best overall performance.
In Table 5.3, the column "Word" means whether the model supports word-level sign language recognition or not. D-G stands for Dynamic Gestures and is indicating if the mentioned models can recognize dynamic gestures. Here, most of the models mentioned here can not construct full-length sentences as output. Our hybrid model can predict isolated sign words and construct sentences to show as output to the users. If we look at Table 5.3, we will also see that our model has the highest accuracy, and it can also recognize sign words that have dynamic gestures. Among the 62 words, there are a large number of words that have dynamic hand movements.

Here, the research work[9] by Tazalli et al. 2022 and the study[12] by R.Sreemathy et al. 2023 both use YOLO as their primary model. We know that YOLO is trained on an image dataset and as a result, they do not support dynamic hand movements. To be more precise, if a sign word has complicated hand movements, the model will not be able to recognize it. Usually, the images of the dataset are annotated manually,

Paper	Model	Classes	Word	D-G	Accuracy
Tazalli et al. 2022	YOLOV5	34	✓	✗	76.29%
Zhang et al. 2024	DPCNN	24	✗	✗	99.52%
Gayathri D. et al. 2024	Mobile-Net V2	26	✗	✗	85.45%
Halvardsson et al. 2020	InceptionV3	26	✗	✗	85%
Kumari et al. 2024	CNN+LSTM	100	✓	✓	84.65%
Paul et al. 2023	CNN+LSTM	29	✓	✗	94.3%
Ma et al. 2022	TSM-ResNet50	29	✓	✓	97.57%
R.Sreemathy et al. 2023	YOLOV4	80	✓	✗	98.8%
Rahman et al. 2021	GANs	20	✓	✗	91.3%
Proposed Model	**CNN+T**	62	✓	✓	**99.58%**

Table 5.3: A comprehensive comparison between our model and the models proposed in earlier research papers

which takes a large amount of time and it is a huge disadvantage. Our hybrid model deals with this problem by using mediapipe for extracting hand landmarks from the videos and labeling the similar landmarks under the same class.
In their research experiments, Paul et al. (2023) [19] and Ma et al. (2022) [8] used a dataset containing 29 different classes. However, they have only 3 words and the rest of the classes are English alphabets. In contrast, we did not include alphabets in our dataset and it has 62 distinct Bengali sign words there. Lastly, the research paper [17] by Kumari et al. 2024 had more word classes (100) in their dataset and the dataset also includes words with dynamic gestures. We have trained the same model proposed in their study with our own Bengali dataset and found that our CNN+Transformers can outperform their model for both training and test datasets. All of the provided data proves that the model we have been working with is the right choice.

Chapter 6

Conclusion and Future Work

In this research paper, our primary objective was to find the most effective hybrid model capable of outperforming more established models in sign language recognition tasks. We wanted to do it to help everyone communicate with deaf people all over the world, especially the ones who use Bengali Sign Language. Although, we have shown with enough evidence that our proposed model can yield exceptional results, there is still much to be done. As we have mentioned earlier, we were focusing mainly on comparing different hybrid and standalone models with the one we have suggested. Hence, we could not allocate time to build an Android app with our trained model. Instead, we have built a web app that takes less time and resources to build. Furthermore, since we lacked an NLP specialist, we were unable to include NLP in this study. Nonetheless, we are pleased with the progress made in this study.

In our next research paper, we are planning to collaborate with someone who excels at Natural Language Processing (NLP) so that we do not have to use conditions to transform our sentences anymore. Furthermore, by using a larger dataset to train our model, we hope to increase its accuracy. We may be able to eliminate the small amount of misclassification our model is showing. In addition, we will also try to bring an Android app developer to make a proper app that we will be able to release on the Google Play Store. This will enable us to help more and more people. We are also hoping to add a large number of new Bengali words to our dataset in order to make our model capable of predicting thousands of Bengali sentences.

In the future, we have plans to work with SLR and SLT that work both ways. To be more precise, we have plans to build an app that will also be able to generate sign language from sentences inserted by users. The app will create animations for different words to construct a complete sentence. For example, if a user inserts the sentence "I will use my phone to talk to you in sign language", the app will be able to show signs for all of these words and create a video which the user can show others to communicate with them. A considerable amount of time and effort are required for building such an advanced system. In conclusion, we hope to be able to create something that can genuinely benefit our society with the knowledge we have acquired while working on this research paper.

Bibliography

[1] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden, *Sign language transformers: Joint end-to-end sign language recognition and translation*, 2020. arXiv: 2003.13830 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2003.13830.

[2] G. Halvardsson, J. Peterson, C. Soto-Valero, and B. Baudry, *Interpretation of swedish sign language using convolutional neural networks and transfer learning*, 2020. arXiv: 2010.07827 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2010.07827.

[3] W. Abdul, M. Alsulaiman, S. U. Amin, *et al.*, "Intelligent real-time arabic sign language classification using attention-based inception and bilstm," *Computers and Electrical Engineering*, vol. 95, no. C, p. 107 395, Oct. 2021, ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2021.107395. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2021.107395.

[4] A. Chaikaew, K. Somkuan, and T. Yuyen, "Thai sign language recognition: An application of deep neural network," in *2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering*, 2021, pp. 128–131. DOI: 10.1109/ECTIDAMTNCON51128.2021.9425711.

[5] A. Elhagry and R. G. Elrayes, "Egyptian sign language recognition using cnn and lstm," *arXiv preprint*, vol. 2107.13647, 2021. [Online]. Available: https://arxiv.org/abs/2107.13647.

[6] M. M. Rahman, R. Mdrafi, A. C. Gurbuz, *et al.*, "Word-level sign language recognition using linguistic adaptation of 77 ghz fmcw radar data," in *2021 IEEE Radar Conference (RadarConf21)*, 2021, pp. 1–6.

[7] A. Tyagi, Ed., *Data Science and Data Analytics: Opportunities and Challenges*, 1st. Chapman and Hall/CRC, 2021, ISBN: 9781003111290. [Online]. Available: https://doi.org/10.1201/9781003111290.

[8] Y. Ma, T. Xu, and K. Kim, "Two-stream mixed convolutional neural network for american sign language recognition," *Sensors*, vol. 22, no. 16, 2022, ISSN: 1424-8220. DOI: 10.3390/s22165959. [Online]. Available: https://www.mdpi.com/1424-8220/22/16/5959.

[9] T. Tazalli, Z. A. Aunshu, S. S. Liya, *et al.*, "Computer vision-based bengali sign language to text generation," in *2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS)*, IEEE, 2022, pp. 1–6.

[10] M. Ghayoumi, *Generative Adversarial Networks in Practice*, 1st. Chapman and Hall/CRC, 2023, ISBN: 9781003281344. [Online]. Available: https://doi.org/10.1201/9781003281344.

[11] K. R. Krishnaiah and A. H. Prasad, "An innovative approach to continuous sign language recognition: Combining cnn and bilstm models with iterative training," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 14, no. 1, pp. 321–333, 2023. DOI: 10.17762/turcomat.v14i1.14004. [Online]. Available: https://turcomat.org/index.php/turkbilmat/article/view/14004.

[12] R. Sreemathy, M. Turuk, S. Chaudhary, K. Lavate, A. Ushire, and S. Khurana, "Continuous word level sign language recognition using an expert system based on machine learning," *International Journal of Cognitive Computing in Engineering*, vol. 4, Apr. 2023. DOI: 10.1016/j.ijcce.2023.04.002.

[13] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, arXiv preprint arXiv:1706.03762, 2023. [Online]. Available: https://arxiv.org/abs/1706.03762.

[14] X. Xu, K. Meng, C. Chen, and L. Lu, "Isolated word sign language recognition based on improved skresnet-tcn network," *Journal of Sensors*, vol. 2023, no. 1, p. 9 503 961, 2023. DOI: 10.1155/2023/9503961. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2023/9503961. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2023/9503961.

[15] U. Bhatti, H. Mengxing, J. Li, S. Bazai, and M. Aamir, *Deep Learning for Multimedia Processing Applications: Volume Two: Signal Processing and Pattern Recognition*. CRC Press, 2024, ISBN: 9781003828051. [Online]. Available: https://books.google.com.bd/books?id=BObpEAAAQBAJ.

[16] G. D., A. Ramar, S. Karpagam, S. J., R. S., and S. P., "Sign language recognition using convolutional neural network," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 17s, pp. 329–337, Feb. 2024. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/4878.

[17] D. Kumari and R. S. Anand, "Isolated video-based sign language recognition using a hybrid cnn-lstm framework based on attention mechanism," *Electronics*, vol. 13, no. 7, 2024, ISSN: 2079-9292. DOI: 10.3390/electronics13071229. [Online]. Available: https://www.mdpi.com/2079-9292/13/7/1229.

[18] S. Nukala, X. Yuan, K. Roy, and O. Odeyomi, "Face recognition for blurry images using deep learning," in *2024 International Conference on Computational Intelligence and Artificial Intelligence (CCAI)*, May 2024, pp. 46–52. DOI: 10.1109/CCAI61966.2024.10603301.

[19] S. Paul, M. A. A. Walid, R. Paul, *et al.*, *An adam-based cnn and lstm approach for sign language recognition in real-time for deaf people*, Apr. 2024.

[20] Y. S. N. Rao, Y. Chong, R. Khan, *et al.*, "Dynamic sign language recognition and translation through deep learning: A systematic literature review," *Journal of Artificial Intelligence*, vol. 102, pp. 7923–7947, Nov. 2024.

[21] S. Richhariya, *Anomaly detection & classification using machine learning for campus surveillance*, Master's thesis, San Jose State University, 2024. DOI: https://doi.org/10.31979/etd.tr8h-menq. [Online]. Available: https://scholarworks.sjsu.edu/etd_theses/5607.

[22] J. Zhang, X. Bu, Y. Wang, H. Dong, Y. Zhang, and H. Wu, "Sign language recognition based on dual-path background erasure convolutional neural network," *Scientific Reports*, vol. 14, no. 1, p. 11 360, May 2024, ISSN: 2045-2322. DOI: 10.1038/s41598-024-62008-z. [Online]. Available: https://doi.org/10.1038/s41598-024-62008-z.

[23] arXiv, *Arxiv: Open-access repository for research papers*, 2025. [Online]. Available: https://arxiv.org/.

[24] Maxapress, *Maxapress: Open access publishing platform*, 2025. [Online]. Available: https://www.maxapress.com/.

[25] Scribd, *Scribd: A digital library of books, audiobooks, and more*, 2025. [Online]. Available: https://www.scribd.com/.

[26] United Nations, *International day of sign languages*, Accessed: 23-Mar-2025, n.d. [Online]. Available: https://www.un.org/en/observances/sign-languages-day.