Bangla Sign Language Recognition: Comparing Deep Learning Approaches using a Custom Dataset

by

Raj Xavier Rozario (173014010)
Md. Abrar Abedin Wohra (182014049)
Siddharth Chakma (181014098)

Capstone project report (CSE 499) submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

Under the supervision of

Khan Raqib Mahmud



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF LIBERAL ARTS BANGLADESH

SPRING 2023

© *Raj Xavier Rozario, Md. Abrar Abedin Wohra,* and *Siddharth Chakma*
All rights reserved

DECLARATION

Project Title	Bangla Sign Language Recognition: Comparing Deep Learning Approaches using a Custom Dataset
Authors	*Raj Xavier Rozario*, *Md. Abrar Abedin Wohra*, and *Siddharth Chakma*
Student IDs	173014010, 182014049, and 181014098
Supervisor	Khan Raqib Mahmud

We declare that this capstone project report entitled *Bangla Sign Language Recognition: Comparing Deep Learning Approaches using a Custom Dataset* is the result of our own work except as cited in the references. The capstone project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Raj Xavier Rozario
173014010

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh

Md. Abrar Abedin Wohra
182014049

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh

Siddharth Chakma
181014098

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh

Date: February 19, 2023

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh
688 Beribadh Road, Mohammadpur
Dhaka - 1207

APPROVAL OF THE COMMITTEE

This is to certify that the CSE 499 capstone project report entitled **Bangla Sign Language Recognition: Comparing Deep Learning Approaches using a Custom Dataset**, submitted by **Raj Xavier Rozario** (173014010), **Md. Abrar Abedin Wohra** (182014049) and **Siddharth Chakma** (181014098) are undergraduate students of the **Department of Computer Science and Engineering** has been examined. We hereby accord our approval of it as the presented work is satisfactory.

Place: Dhaka
Date: February 19, 2023

Khan Raqib Mahmud
Senior Lecturer

Muhammad Golam Kibria, PhD
Professor & Acting Head

Nafees Mansoor, PhD
Associate Professor

Mohammed Ashikur Rahman, PhD
Assistant Professor

ACKNOWLEDGEMENTS

We would like to express our deep and sincere gratitude to our research supervisor, *Khan Raqib Mahmud*, for giving us the opportunity to conduct research and providing invaluable guidance throughout this work. His dynamism, vision, sincerity and motivation have deeply inspired us. He has taught us the methodology to carry out the work and to present the works as clearly as possible. It was a great privilege and honor to work and study under his guidance.

We are greatly indebted to our honorable teachers of the Department of Computer Science and Engineering at the University of Liberal Arts Bangladesh who taught us during the course of our study. Without any doubt, their teaching and guidance have completely transformed us to the persons that we are today.

We are extremely thankful to our parents for their unconditional love, endless prayers and caring, and immense sacrifices for educating and preparing us for our future. We would like to say thanks to our friends and relatives for their kind support and care.

Finally, we would like to thank all the people who have supported us to complete the project work directly or indirectly.

Raj Xavier Rozario, *Md. Abrar Abedin Wohra* and *Siddharth Chakma*
University of Liberal Arts Bangladesh
Date: February 19, 2023

Dedicated to
my adoring Mother.

– Raj Xavier Rozario

To my beloved parents
the source of my inspiration.

– Md. Abrar Abedin Wohra

To my father
for his never-ending motivation.

– Siddharth Chakma

ABSTRACT

The rate of disability in Bangladesh is increasing every year and every year these people cry out for recognition. The impaired have been demanding more awareness and education in the field of sign language. Translating signs using motion-based deep learning is an evolving field. Our focus was to assimilate the largest word level dataset and apply that to create a system that could be used in the recognition of Bangla Sign Language. To further develop this, we increased the data and class variations through different techniques. We trained three different models with the dataset to measure the degree of accuracy between the models in detecting the signs. Modifications were also made to one of the models to actuate performance improvements. The comparison between these three models led to an analysis that indicates that the graph based model outperformed the others. With further additions to variety and samples for each class, practical implementation could be made to translate signs in real time to help bridge the communication troubles between us and the hearing/speaking impaired.

Keywords: Sign Recognition, Custom Dataset, BdSL, I3D, TGCN, C-RNN

Contents

1 Introduction **1**

1.1 Problem Statement 1

1.2 Aims, Objectives and Motivation 2

1.2.1 Aims 2

1.2.2 Objectives 2

1.2.3 Motivation 3

1.3 Project Specification 3

1.3.1 Cost Analysis 4

2 Literature Review **6**

2.1 Research in BdSL 6

2.2 Word Level Research 7

3 Methodology **9**

3.1 Data Collection 10

3.2 Pre-Processing 12

3.3 Data Augmentation 13

3.4 Data Annotation 14

3.4.1 OpenPose Architecture 15

3.4.2 Extraction 16

3.4.3 Labelling 17

3.5 Data Arrangement 18

3.6 Novelty 19

4 Design and Development **20**
4.1 Inflated 3D Network (I3D) 20
4.1.1 Inflated 3D Network Architecture 21
4.2 Temporal Graph Convolutional Network (TGCN) 22
4.2.1 Temporal Graph Convolutional Network Architecture 23
4.3 Convolutional-Recurrent Neural Network (C-RNN) 24
4.3.1 Convolutional-Recurrent Neural Network Architecture 24
4.4 Training 26
4.5 Web Application 28

5 Result Analysis **30**
5.1 Performance Metrics 30
5.1.1 Confusion Matrix 30
5.1.2 Evaluation Metrics 31
5.1.3 Functions 31
5.2 TGCN Performance 32
5.3 I3D Performance 35
5.4 C-RNN Performance 35
5.5 In Comparison 36
5.6 Webcam Testing 37

6 Conclusions **38**
6.1 Legal and Ethical Issues 38
6.2 Social Impact 39
6.3 Conclusion 40

References **41**

List of Figures

3.1 Project Flowchart. 9
3.2 A page from the curated word list. 10
3.3 Deaf students at the BNFD Deaf Institute. 11
3.4 Original data representing the words: I - Want - Food 12
3.5 Three Additional Signers 13
3.6 Different Applied Augmentations 14
3.7 OpenPose Architecture 15
3.8 Frame by Frame Pose Extraction. 16
3.9 File Arrangement after Renaming. 17
3.10 Different Label Structures for Different Models. 18

4.1 The I3D Architecture 21
4.2 Inside an Inception Module 22
4.3 The TGCN Architecture 23
4.4 MobileNetV2 Architecture 24
4.5 Single and Bidirectional LSTM Architecture 25
4.6 C-RNN Architecture 26

5.1 Batch A Accuracy and Loss Comparison. 32
5.2 Accuracy and Loss Comparison between Batches A and B. 33
5.3 Accuracy and Loss Difference for 100 words. 33
5.4 Confusion Matrix Comparison for 100 words. 34
5.5 Accuracy and Loss Comparison between Batches B and C. 36
5.6 Webcam Test 37

6.1 Consent Form 38
6.2 Payment Receipt 39

List of Tables

1.1 Cost Analysis 5

3.1 Dataset Distribution 18

4.1 Training Parameters 27

5.1 TGCN Performance Overview 34
5.2 TGCN Evaluation Overview 35
5.3 I3D Evaluation Overview 35
5.4 C-RNN Evaluation Overview 36
5.5 Model Accuracy Comparison 37

Chapter 1

Introduction

1.1 Problem Statement

Sign language is the only mode of communication for those impaired of hearing and speaking. Over 2.4% Tajmim (2022) of the population of Bangladesh was diagnosed with disabilities last year alone and that number is increasing annually. Being unable to hear or speak hampers the effectiveness of their communication with other people which in turn is the main cause of job rejections and unemployment Jahan (2019). Education in sign language is important now more than ever as there are very few institutions Sakib (2019) in Bangladesh that specialize in teaching sign language to people. Illustrated books are available for those willing to learn but static pages cannot enunciate the dynamic nature of these signs. Another issue is that these books are rarely made available to the public and remain exclusive to institutes that study or teach this language. Books that are available date back to the early 2000s and therefore are quite outdated Karim (1994) as the language is being frequently revised. The disabled cannot effectively communicate with other individuals with signs alone as they need trained interpreters to translate for them. Most disabled people in Bangladesh are from poor backgrounds and cannot afford an interpreter therefore sticking to communicating only amongst themselves.

Research in Bangladeshi Sign Language (BdSL) has mostly been image based Rafi et al. (2019) and on an alphabetical level Fahad et al. (2020) with almost no word level work done. Word level signs are performed through the movement of the hands, but they also are dependent on their interactions with the various parts of the body to express more complex words. A visual element sometimes accompanies the language as signers often use facial expressions to convey the emotional emphasis

they want to associate with the sign. Not everyone signs the same way and slight deviations often bring a change in meaning. Due to their dynamic nature, sign gestures need to be tracked over time to classify and recognize them as these signs would not work as static images which is a great impediment. BdSL neither has a proper extensive word level data-set nor one in video format. Therefore, the target problem of this project is curating a large video data-set consisting of words used frequently in Bengali Sign Language and then implementing it on three different types of deep learning models that can process entire sequences of gestures and then analyse which method performs the best.

1.2 Aims, Objectives and Motivation

1.2.1 Aims

We set our aim to develop a word level video dataset based on the Bangladeshi Sign Language and present three different computer vision-based approaches for its classification and recognition. Conventional sign datasets in Bangladesh are static images of alphabets with words rarely included. We would solely focus on commonly used words in the Bangla language and the size of the dataset which would be the largest in existence for BdSL in terms of words.

For training, we aimed to conduct an experiment with three distinct types of deep learning models. A 3D CNN, a Graph-based model. We also aim to modify a CNN-RNN model to determine which RNN component is most suitable. Based on the results, we would conduct a comparative performance analysis of all models on different subsets of the dataset. The best model can be later used to translate signs on hardware systems, thus establishing a digital form of communication between us and the disabled here in Bangladesh.

1.2.2 Objectives

Based on aim, the following were the objectives we set for the project:
1. Find an authentic data source for the dataset.
2. Make a fixed word list to capture.
3. Record videos from the professional.
4. Pre-process and annotate the data.

5. Find suitable contrasting models for training.
6. Experiment with a CNN-RNN model.
7 .Analyze and compare training and test scores.
8. Create a system to recognize sign gestures.
9. Bridge the communication gap between us and the disabled.

1.2.3 Motivation

Sign Language is a visual form of expression for those who cannot hear or speak and are natural languages complete with their own semantics, vocabulary, and grammar. These are generally modeled after the surroundings and culture of individual regions. BdSL has been developed over many years encompassing over 3000 signs for alphabets, numerals, and words. Several books have been published transcribing signs in illustrations for education. Despite this, there is a significant disconnect in communications between us and the disabled. This creates an incredible challenge for them to stay involved in this competitive world. Unemployment and social isolation are some of the effects. Researchers have tried to employ deep learning to automatically interpret sign language with the majority being done on ASL. For BdSL, there is a research void in recognition in the video space which needs to be explored.

1.3 Project Specification

This section lists all the requirements that this project entails into different areas related to both data-set creation and model training. The cost analysis section gives an overview of the expenses that accompany those requirements.

The prime requirement considering usability in translation was to create a balanced sign language data-set which was precise and valid. To fulfill that an expert is needed to faithfully perform those signs and later validate them. Due to the nature of the project and the high volumes of data needed, powerful equipment is vital for training.

Hardware Specifications:
CPU: 2x Intel Xeon @2.30 GHz
GPU: NVIDIA Tesla T4 16 GB
Minimum RAM: 32GB
Minimum Disk Space: 120GB
Video Camera
62cm Tripod

Camera Specifications:
Camera Resolution: 48 MP
Video Resolution: 2336 x 1080
Aperture: f/2.0
Field of View: 80 degrees
Framerate: 30 Frames per second

Component Specifications:
Davinci Resolve 18
Google Colab Pro+
Microsoft Excel
OpenPose
Python

Tensorflow
PyTorch
CUDA and cuDNN
Matplotlib
OpenCV and others.

1.3.1 Cost Analysis

The implementation of this project necessitated going through several steps with the most important being the collection of the dataset. People usually make use of existing online datasets or take pictures from their webcams for signs. We had to create a dataset of our own to match the scope of the project that required the assistance of an expert. The expert provides his valuable knowledge and performance for the dataset but not for free. As students, this was distressing news as we scurried to gather the amount from borrowing it from our guardians and our own pockets. The situation also came with legal challenges which are documented in Chapter 6.Traveling back and forth during this collection phase also incorporates travel fares. To process and train large volumes of video data, a powerful hardware setup is prerequisite. While local training was possible, due to factors such as constant blackouts, incompatibility and lack of raw processing power, we had to subscribe to Google Pro Plus in the initial stages of training. However, the IDEs and the libraries used here were open source and free. Deploying a software based on this project would involve a significant cost during launch and after in maintenance and bug fixing.

Table 1.1: Cost Analysis

Mandatory Components	Cost
Expert Dataset	20,000
Phone/Camera	21,000
Tripod	1,000
Travel Expenses	3,000
Google Pro +	6,000
Storage	5,000
Future Components	
Deployment	10,000
Maintenance	4,000
Additional Data	15,000
Total Cost	85,000

To get better results, the data can always be increased and so can the classes therefore that would require getting more signers and holding additional shooting sessions. The entire dataset would be voluminous and would demand storage on the cloud or portable drives. Table 1.1 above gives an analysis of every necessary and future costs related to this project.

Chapter 2

Literature Review

2.1 Research in BdSL

Research in BdSL has been ongoing for some time leading to multiple different outcomes that mostly deal with detection on an alphabetical or numerical level. Urmee et al. (2019) made use of the Xception model as a base for detecting 37 alphabets. They developed their own BdSLInfinite dataset which they trained with transfer learning which reached a test accuracy of 98.93%. Later, Podder et al. (2022) aimed to use semantic segmentation to mask the hand data to perform a comparative analysis on 3 pre trained CNN models. ResNet18, MobileNetV2 and EfficientNet all managed an overall accuracy of 99.05%. Segmentation was also utilized by Sadik et al. (2019) in combination with binary masking using Fuzzy C-means clustering to get features from 10 alphabets. These are later trained with a multi class SVM model for classification. The CubicSVM model was tested for 99.50% accuracy.

Hossen et al. (2019) attempted to tune the layers of the VGG16 model to detect signs only to achieve 84.68% in validation. Talukder & Jahara (2020) proposed a way to use object detection to interpret signs from various sequences of letters and signs and string them together to generate sentences in form of text. Training using the YOLOv5 model, they achieved an accuracy of 97.95%. Three different datasets BdSL, KU-BdSL and Ishara-Lipi were segmented using HSV and watershed algorithms then augmented by Miah et al. (2022). Training took place on a CNN model with modified hyperparameters called BenSignNet. Each dataset ended up with an accuracy of 94.00%, 99.60% and 99.60% respectively. In a very recent study in BdSL by Lipi et al. (2022), a small dataset of 30 words made up of static images

was trained on a CNN model using the Adam optimizer generating an accuracy of 92.50

2.2 Word Level Research

On the word level, perhaps the biggest is the research based on the WLASL dataset by D. Li et al. (2019). Consisting over 3,126 words performed by 100 signers, the dataset was split and tested on 3 different models: VGG-GRU, I3D and Pose-GRU. All these models are tuned to capture both the spatial and temporal data from video frames with Pose-GRU using OpenPose to extract keypoints as input features. Each model achieves an average accuracy of 47.6%, 79.3% and 56.3% based on top-K classification. Based on the latter and LSA64 datasets, Boháček & Hrúz (2022) proposed a transformer model which uses VisionAPI to retrieve body landmarks and normalizes them for input. The model is trained using the SPOTER architecture with an SGD optimizer resulting in a test accuracy of 63.18% for WLASL and 100% for LSA64. Continuing with WLASL, Hosain et al. (2021) articulated a video-based method that combines pose pooling with different 3D Convolutional networks in an attempt to improve performance on the different subsets of WLASL, achieving only a 4% increase. Alternatively, Maruyama et al. (2021) offered a multi-stream method to simultaneously process body images and pose data on a modified I3D model. When tested on the WLASL dataset, this method provides a 15% performance increase over previously tested techniques.

Jiang et al. (2021) proposed an alternative to standard skeleton-based approach called Skeleton Aware Multi-modal Framework with a Global Ensemble Model (GEM). It consists of a Sign Language Graph Convolution Network (SL-GCN) and a Separable Spatial-Temporal Convolution Network (SSTCN) to process skeleton features. These are fused with GEM to create accurate predictions. Similarly, de Amorim et al. (2019) used a Spatial Temporal Graph Convolution Network which allows for graphs to get both spatial and temporal features from the signs. Working on a 20-word dataset, it reaches an accuracy of 61.04%. Vo et al. (2019) collected two datasets 12 to 15 words and separated the signing region through skin segmentation. The segmented data is trained using both a static image approach through SVM and temporal approach through a pre-trained VGG16 and LSTM for prediction. The KETI (Korea Electronics Technology Institute) dataset which boasts 14,672 high

resolution videos from 10 signers was presented by Ko et al. (2019). The sequence-to-sequence translation model used achieves an accuracy of 93.28% but the dataset still lacks variation in environments. In contrast, Rahman et al. (2021) used 77 GHz RF (Radio Frequency) sensors to capture the motion of signers to create a small dataset of 20 words. These are processed through GANs (Generative Adversarial Networks) with micro-Doppler signatures to increase performance. WGAN has an accuracy of 74.28% while MBGAN has 80.82%. After boosting, they have scores of 86% and 91.30% respectively.

While deeper dives have been taken into temporal segmentation D. Li et al. (2020) and other feature improvements but Bangladeshi datasets have not been able to successfully address word level recognition because of their insistence on staying static due to the performance and convenience of image-based recognition. The literature reviewed here has given us valuable insight into dataset creation, processes to extract pose features. Additionally, it also provided a qualitative comparison of the various models that could be used for our project.

Chapter 3

Methodology

In this chapter, we discuss our initially planned workflow and the processes that led to the creation of our dataset. We also take a look into the pose extraction process, various data augmentations and the concluding dataset arrangement. The flowchart in Figure 3.1 below provides a methodical illustration of the project pipeline.

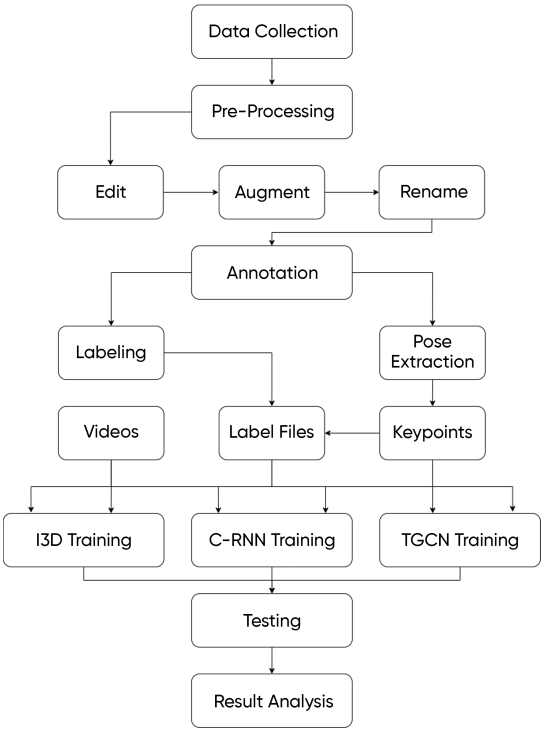
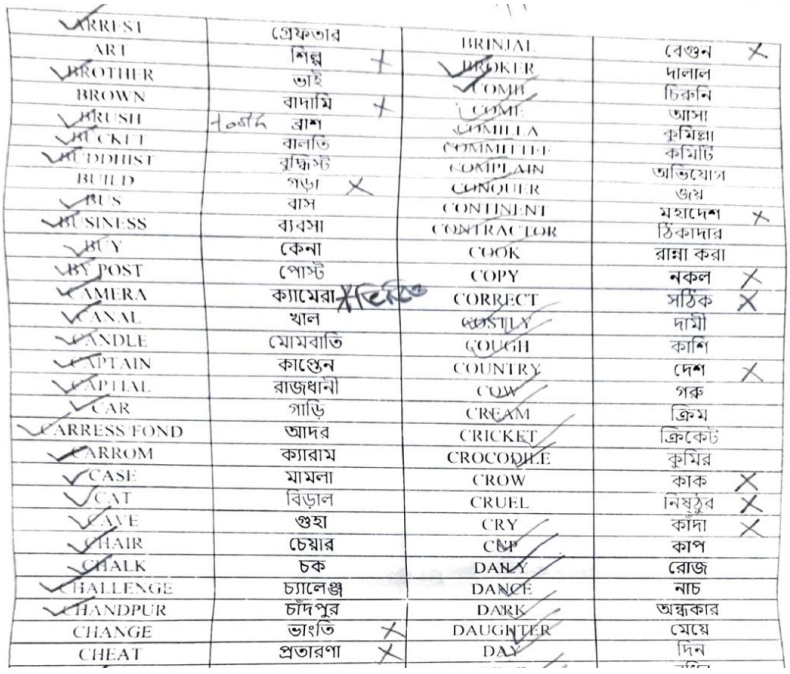


Figure 3.1: Project Flowchart.

3.1 Data Collection

In this section, we discuss the planning and execution behind creating our own unique dataset. While planning, we decided to focus on authenticity, sign accuracy and a collection of words that are used daily. We targeted a list of 1000 frequently used words for inclusion seen in Figure 3.2. One of the earliest limitations we faced was the lack of datasets and reference material that aligned with our aim for the project. The only substantial material we found was a sign dictionary dating back to 1994. To overcome this, we attempted to create a custom dataset and therefore, we required an official source to collect the data from and finding a sign institute was difficult as they are scarce.



ARREST	গ্রেফতার	BRINJAL	বেগুন
ART	শিল্প	BROKER	দালাল
BROTHER	ভাই	TOMB	চিরুনি
BROWN	বাদামি	COME	আসা
BRUSH	ব্রাশ	COMILLA	কুমিল্লা
BUCKET	বালতি	COMMITTEE	কমিটি
BUDDHIST	বুদ্ধিস্ট	COMPLAIN	অভিযোগ
BUILD	গড়া	CONQUER	জয়
BUS	বাস	CONTINENT	মহাদেশ
BUSINESS	ব্যবসা	CONTRACTOR	ঠিকাদার
BUY	কেনা	COOK	রান্না করা
BY POST	পোস্ট	COPY	নকল
CAMERA	ক্যামেরা	CORRECT	সঠিক
CANAL	খাল	COSTLY	দামী
CANDLE	মোমবাতি	COUGH	কাশি
CAPTAIN	কাপ্তেন	COUNTRY	দেশ
CAPTIAL	রাজধানী	COW	গরু
CAR	গাড়ি	CREAM	ক্রিম
CARRESS FOND	আদর	CRICKET	ক্রিকেট
CARROM	ক্যারাম	CROCODILE	কুমির
CASE	মামলা	CROW	কাক
CAT	বিড়াল	CRUEL	নিষ্ঠুর
CAVE	গুহা	CRY	কাঁদা
CHAIR	চেয়ার	CUP	কাপ
CHALK	চক	DAILY	রোজ
CHALLENGE	চ্যালেঞ্জ	DANCE	নাচ
CHANDPUR	চাঁদপুর	DARK	অন্ধকার
CHANGE	ভাংতি	DAUGHTER	মেয়ে
CHEAT	প্রতারণা	DAY	দিন

Figure 3.2: A page from the curated word list.

That book we found was instrumental in our institutional scouting process. It was published by the Bangladesh National Federation of the Deaf (BNFD) pictured in Figure 3.3 which we decided to visit after a lengthy search. After a lengthy troublesome process of creating paperwork, canceled appointments, the Director of the BNFD, agreed to collaborate with us. Lion Manwer Hossain, the director, was a professional television signer and the foremost sign translator in Bangladesh.

Following prolonged deliberation and government bureaucracy, an authorized letter was provided to him on behalf of ULAB, we came to terms on remuneration and he officially became our primary data source.



Figure 3.3: Deaf students at the BNFD Deaf Institute.

Multiple recording sessions were set up that would span to different times of the day so that we would have variety in lighting but due to the signer's scheduling issues on weekdays, we were limited to shoot only on weekends. Coordinating with the signer's availability and ease of location, we had to travel every weekend to Paltan for the shoot. Since we cannot afford a studio and the institute lacked a controlled environment to record, we had to relocate to the roof which led to further complications. The sessions took place under intense heat followed by heavy rain and sometimes thunderstorms where despite our enthusiasm to continue, we were forced to cancel by the signer. The RGB videos were shot on a 62 cm tripod setup with a phone at 60 frames per second with a resolution of 2340 x 1080p. During recording, upon surveying the word list, many of the words unfortunately had to be omitted due to them being a focus of future revision or not being in use anymore. Thus, we ended up recording videos of about 733 words out of the 1000 targeted from the signer who performed each word once. These videos were shot in front of a clean single colored background to keep it viable for removal if required by potential augmentations Cauli & Recupero (2022).

The accumulated dataset was thoroughly scrubbed for any signs that were similar in motion or meaning. Filtering out these homogeneous signs was necessary to ensure that the dataset only has unique signs, and the model only extracts

relevant data during training. The videos were 15 to 20 minutes each and had to be meticulously cut and split into several hundred different videos with an average length of 2 seconds. We ended up with 700 videos of unique word signs to use both as data and reference as seen in Figure 3.4.



Figure 3.4: Original data representing the words: I - Want - Food

3.2 Pre-Processing

In this section, we discuss the pre-processing phase which is essential in creating a balanced dataset and improving data quality which in turn refines model performance. Pre-processing also allows us to keep a consistent resolution for each batch and remove any redundant frames. Before training, the videos had to be scaled down into different resolutions for each model. To keep clarity for the pose extraction process, the videos were first processed from .MOV to .mp4 at a 1920 x 1080p resolution. As per the requirements of the CNN based models, the videos were re-rendered at a 256 x 256 resolution. The framerate of the videos was also reduced to 30 fps for both instances.

To test out an augmentation for the pose-based method and to increase the robustness of the dataset, these videos were cropped to various perspectives that would emulate common user camera placements. This increased the dataset size

to 2100 videos of 700 glosses. The cropped perspectives position both the body and sign differently. The test was to see whether the extracted poses would act as distinct data during training or would over-fit the model. All video processing was carried out using Davinci Resolve 18.



Figure 3.5: Three Additional Signers

Having three variations of the same data is not enough to train the models, so using the original set as a reference, three additional signers as seen in Figure 3.5 recorded the first 200 signs under different conditions such as lighting and background. Due to the time consuming process or recording and pre-processing, only 200 signs were recorded. These 600 videos were then passed through the pre-processing pipeline described above. The dataset now consists of 2,700 videos with signs from 4 different signers. The total length amounts to exactly 1 hour 52 mins and 14 seconds where each video ranges from 1 to 3 seconds.

3.3 Data Augmentation

After the preliminary training sessions with all the models, it was observed that even the data from the additional signers was not enough, so we decided to augment the data. Data Augmentation is generally used to increase the size and variety of the training data which aids the model in generalizing with unseen data and avoiding over-fitting. We handpicked the types of augmentations to apply as seen in Figure 3.6 and used OpenCV to process the videos and create the augmented data. The clockwise, anti-clockwise rotations, flipping, horizontal and vertical shifting were selected because we could not have data remain in the same position due to the key-point tracking process. To get different data for both the image based and pose

based model without having separate sets of augmentations, a compromise was made and other techniques such as brightness, Gaussian blur and noise were not considered. We hoped that by exposing the model to these augmentations, it would

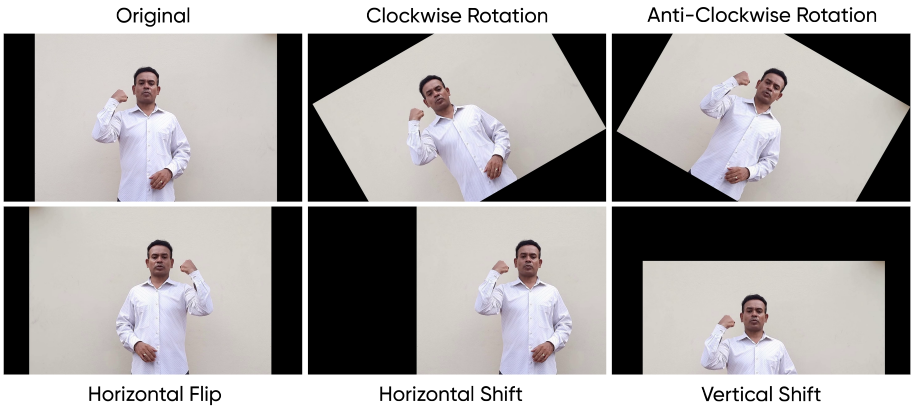


Figure 3.6: Different Applied Augmentations

help it learn how to recognize these signs from different angles. After augmentation, we had 20 additional samples for each word amounting to a total of 4000 extra videos for 200 words.

3.4 Data Annotation

In this section, we discuss the different annotation and labelling processes for the models used. The annotation process was time-consuming but also crucial for the performance of the model. The more precise and consistent the annotations are, the better the model will perform. Due to being unfamiliar with the key point extraction technique, it required some time familiarizing ourselves with the tool and the labeling process for each model.

Pose based models operate on three dimensions which require them to process a sequence of poses over a period of time in order to classify motion. Poses are made up of interconnected skeletal key-points whose motions are used by the model as temporal features. This annotation process consists of pose extraction through OpenPose Cao et al. (2018) which is a pose estimation system that can simultaneously recognize and track different human body parts in the form of key-points. OpenPose recognizes a total of 135 key-points consisting of the face,

body, hands, and feet. It allows for both video and image processing therefore it was suitable for our video dataset.

3.4.1 OpenPose Architecture

The OpenPose architecture in Fig. 9 below is defined by two iterative prediction stages, where the first predicts affinity fields and the second predicts confidence maps Noori et al. (2019). To improve upon the previous model Cao et al. (2016) and reduce computation, the 7x7 kernels are replaced with 3 back-to-back 3x3 convolutional kernels whose outputs are concatenated G. Li et al. (2020).

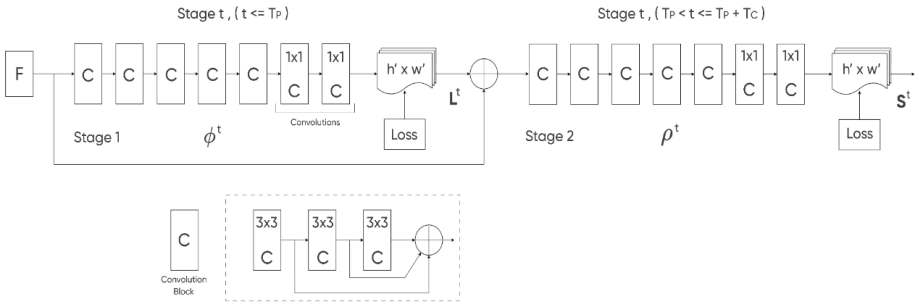


Figure 3.7: OpenPose Architecture

First, we input our videos into the OpenPose pipeline. These videos are processed frame by frame by a CNN made from the first 10 layers of VGG-19 Meena et al. (2022). This process generates feature maps F which are input to the first stage network that predicts Part Affinity Fields (PAFs) that are refined over phases using previous predictions and base features The second stage takes the latest PAF output after a number of iterations, and the process is repeated to refine confidence map predictions.

Confidence Maps Tian et al. (2022) are grayscale images that represent the probability of a body part being present in any pixel. These maps S are defined by the equation 3.1 below where J is the total count of parts.

$$S = (S_1, S_2..., S_J) \; where \; S_j \epsilon R^{h*w}, \; j \epsilon 1...J \tag{3.1}$$

Part Affinity Fields Zhou & Li (2019) create a relationship between the different parts of the same body, so they are mapped accurately for each individual in the video frame. These 2D vector fields save both the orientational and locational

information of each body part while also connecting closely associated ones by creating unit vectors that point to each part in a limb. This is defined by the equation 3.2 below where C is the total count of part connections.

$$L = (L_1, L_2..., L_C) \; where \; L_c \epsilon R^{h*w*c}, \; c \epsilon 1...C \tag{3.2}$$

An L2 Loss Function Kornblith et al. (2020) is utilized to compute the loss between ground truth and predicted affinity fields and confidence maps. The loss functions of both modules at stage t are combined to form the overall equation 3.3 below where L and S represent the ground truth affinity fields and confidence maps.

$$f = \sum_{t=1}^{T_P} f_L^t + \sum_{t=T_P+1}^{T_P+T_C} f_S^t \tag{3.3}$$

3.4.2 Extraction

The maps and fields are passed through a bipartite graph matching algorithm van den Brand et al. (2020) which follows a greedy approach to interconnect each body part and visually represent the key-points of each individual in the form of poses. These poses as seen in Figure 3.8 are full body skeletons consisting of 15 body and 42 hand key-points (21 for each hand).



Figure 3.8: Frame by Frame Pose Extraction.

Each body part has a different module assigned to produce key-points. Since most signs are performed above waist, we disabled the leg connections module before annotating. The key-points are written into arrays consisting of (x, y, c) coordinates where (x, y) represents body coordinates on their respective axes and (c) the detection confidence. To act as input features in a model, the arrays are saved as .JSON files for each frame of the video and renamed to match with their respective videos. This method of annotation focuses only on each subject's movements and sign gestures while removing the factors of the subject itself and its background altogether.

3.4.3 Labelling

We first create a class list to keep track of every gloss being used for each set. Each video in the dataset was titled according to their gloss but to keep them serialized each sample was renamed to assign them a numerical ID as seen in Figure 3.9.

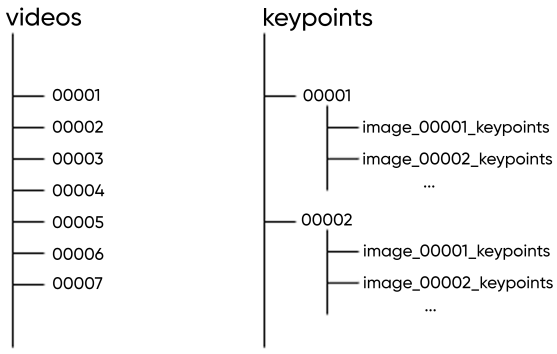


Figure 3.9: File Arrangement after Renaming.

Frame markers are then extracted to specify the frames where the sign motion begins and ends. Each signer was also assigned a designation to distinguish each variation. This also helps in keeping track of the variations being allocated for training, validation and testing. We compiled all the information above into a .csv file which we edited in Excel where each gloss pointed to its video samples and their metadata. We wrote a script to read and parse the data from the .csv file and write it in .JSON according to the formats that complement the models. As seen in Figure 3.10, the action label was created for the I3D model, where the action label itself refers to the gloss position in the class list, start and end frame respectively.

Action Label


```
{
    "00001": {
        "subset": "train",
        "action": [
            0,
            0,
            55
        ]
    },
    "00205": {
        "subset": "val",
        "action": [
            0,
            0,
            60
        ]
    },
```


Data Label


```
{
  "gloss": "age",
  "instances": [
    {
      "fps": 30,
      "frame_end": 50,
      "frame_start": 0,
      "instance_id": 0,
      "signer_id": 1,
      "split": "train",
      "video_id": "00019"
    },
    {
      "fps": 30,
      "frame_end": 52,
      "frame_start": 0,
      "instance_id": 1,
      "signer_id": 2,
      "split": "val",
      "video_id": "00223"
    },
```


Figure 3.10: Different Label Structures for Different Models.

The data label is used universally for all the models to read the classes, frame data and split order from.

3.5 Data Arrangement

With the dataset compiled, we wanted to experiment with different batches of data and see how they would perform on all models. So, we separated the dataset into batches with varying numbers of glosses and signers as seen in Table 3.1 below.

Table 3.1: Dataset Distribution

Batch	Model Used	Glosses	Videos	Signers
A	TGCN	700	2100	1
B	TGCN, I3D, C-RNN	200	1200	4
C	TGCN, C-RNN	200	4800	4

The batches were divided into different subsets for training and comparison with each division having discrete gloss numbers to evaluate the effect of sample sizes and class inflation on recognition. Batch A = {100, 200, 400, 700} contains

samples from the original signer, Batch B = {100, 200} has samples from the latter and 3 additional signers while Batch C = {100, 200} consists of augmented samples from all signers. Glosses have a uniform number of samples for all batches. Batch A has 3 samples for each gloss, Batch B has 6 while Batch C has 24. We then split each sample in a gloss for training, validation and testing in a ratio of 80:10:10 respectively for all models.

3.6 Novelty

An absence of word-level research in BdSL resulted in no proper datasets. Our video dataset with its scale, modernization and authenticity in data is the first of its kind. We trained this dataset on three different kinds of models one of which was a CNN-RNN model we modified ourselves to give improved results.

Chapter 4

Design and Development

This chapter is where we take a detailed look into the models we will be using and how they differ from each other. We also give an overview of the training process, the parameters being employed in their respective training subsections and the resultant product of the project.

4.1 Inflated 3D Network (I3D)

Inflated 3D Network (I3D) is a 3D CNN model Carreira & Zisserman (2017) built upon a Two-Stream 2D CNN Szegedy et al. (2015) model which processes video frames by taking the average predictions from individual RGB frames after passing them through a CNN pre-trained on ImageNet Sharma et al. (2021). The Inception V1 model was modified by inflating all its pooling kernels and filters which allows it to process the temporal features along with the spatial features from each frame. Filters which are traditionally in square (N x N) format turn into (N x N x N). 3D filters are bootstrapped from 2D filters already pre-trained on massive datasets. The filters are then fine-tuned using Kinetics Kay et al. (2017) to improve feature extraction on videos.

Convolutions and Pooling in 2D CNNs are symmetrical which means they concentrate only on the height and width of images. A receptive field Araujo et al. (2019) is a piece of an image that can be seen by one filter at a time, and it increases as more layers are stacked. To incorporate Time, an optimal receptive field which relies on image dimensions and frame rate is necessary. If the receptive field grows too slow in time relative to space, it does not properly record scene dynamics and if it grows too fast, it might combine edges from different objects which disrupts

early feature detection. Therefore, to process temporal features, the kernels in the I3D model are asymmetrical.

4.1.1 Inflated 3D Network Architecture

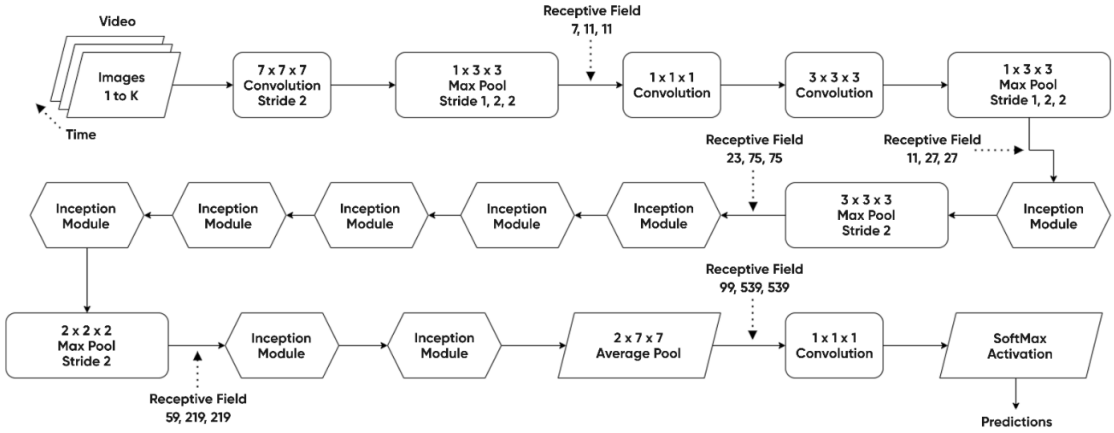


Figure 4.1: The I3D Architecture

The I3D architecture in Figure 4.1 can be described in 3 stages: In the first stage, the videos are taken as input in the form of frames over time. Asymmetrical filters being used for max pooling maintain time while performing Spatial pooling. Low-level spatio-temporal features are extracted by the 3-dimensional convolutional layers.

The second stage consists of stacked Inception modules Hegde & Gangisetty (2021) that have multiple branches, as seen in Figure 4.2, that allow for the use of convolutional filters of various sizes in a single image block. This enables the network to grow wider and process more complex features. Each branch extracts spatio-temporal features at different scales. The (1 x 1 x 1) convolutions are used to decrease the number of input channels prior to the (3 x 3 x 3) convolutions thus rendering it computationally cheaper. The outputs from the filters are then concatenated and passed onto the next layer.

The last stage is where classification takes place. The linear classifier includes a fully connected layer succeeded by a SoftMax Activation Function. Predictions made convolutionally in time in previous layers are averaged and the final class scores are produced.

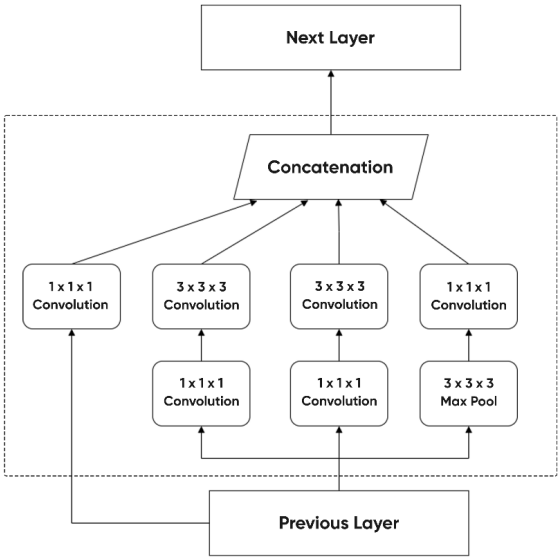


Figure 4.2: Inside an Inception Module

Every convolutional layer except the last is succeeded by a single batch normalization layer and ReLU activation function which are not displayed in the diagram. Each pooling layer is followed by a Receptive field presented in the (time, x, y) format where the units are frames and pixels.

4.2 Temporal Graph Convolutional Network (TGCN)

Graph Convolutional Network (GCNs) Kipf & Welling (2016) are neural networks designed to analyze data that are represented as graphs. They model the progress of the graphs over time and have been used in tasks such as traffic forecasting Zhao et al. (2018), social influence Wu et al. (2018) and motion kuang Chiu et al. (2018) predictions. Temporal Graph Convolutional Network (TGCN) is originally based on GCN which uses filters to extract features from the graph nodes and edges. The difference is that TGCN can model both the spatial and temporal features from a series of poses. Instead of modeling the motion of individual joints, this model considers the motion of the entire body by modeling the paths that the key-points on the body follow over time. This averaged rendition of motion information provides a more accurate representation compared to traditional pose models.

4.2.1 Temporal Graph Convolutional Network Architecture

The first layer receives the coordinates of the body key-points in the form of K x 2N matrix. Bodies are perceived as interconnected graphs with K vertices and edges in the form of weighted adjacency matrix $A_i \in R^{K*K}$ where the weights represent the dependencies between joints. In a standard GCN, each layer is a method G_N that takes as input a feature matrix $H_n \in R^{K*F}$ where F is the feature dimension produced by the previous layer. The graph convolutional layer Shi et al. (2018) can be expressed as the equation 4.1 below where A_n is the trainable adjacency matrix, σ is the nonlinear activation function tanh and W_n are the trainable weights sets $W_n \in R^{F*F'}$.

$$H_{n+1} = G_n(H_n) = \sigma(A_n H_n W_n) \tag{4.1}$$

Graph Convolutional blocks She et al. (2021) residually stack two GC layers where the output of each layer is used as input of the next layer and the process is repeated until a satisfactory accuracy is achieved. Creating a residual connection allows the network to easily refine data and eliminate the cost of modeling it from scratch at every layer.

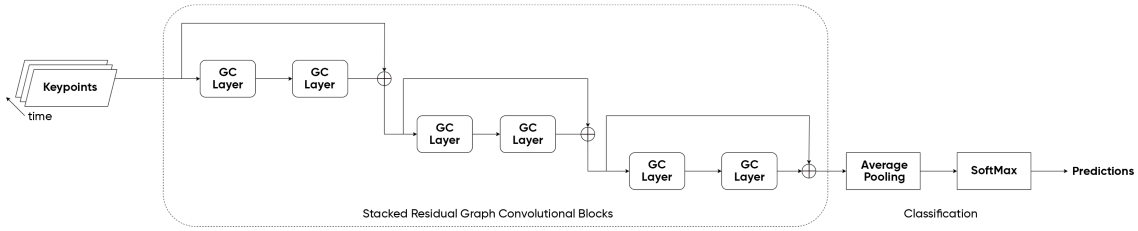


Figure 4.3: The TGCN Architecture

As seen in Figure 4.3, TGCN instead stacks multiple residual GC blocks in succession and uses the average pooling results produced over time as features of the key-point trajectories. Not shown are Batch Normalization layers that appear after every GC layer and dropout layers that appear at the end of each block. The stacking allows the model to learn more complex relationships between different gestures. By averaging data over time, the pooling results capture only the relevant aspects of the tracked motions while discarding the irrelevant. A SoftMax Activation layer accompanied by an Average Pooling layer carries out the classification task.

The SoftMax layer transforms the features into probability distribution over different classes while the Average Pooling layer aggregates those probabilities over time.

4.3 Convolutional-Recurrent Neural Network (C-RNN)

The combination of a 2D CNN and RNN Yin et al. (2019) to extract spatial features of input images and the temporal dependencies within them is common. Here, we breakdown how we modified an existing Convolutional-Recurrent Neural Network (C-RNN) to fulfill our requirement of having an efficient training process where the produced model would be compatible to be used in live detection while also providing better accuracy.

4.3.1 Convolutional-Recurrent Neural Network Architecture

Architecturally, the first part of the existing model is the input layer where videos passed through. A frame generator extracts frames from these videos and samples them. During sampling, the frames are divided into parts and the frames where no signs are performed are eliminated. After sampling, videos are made up of 10 frames and due to videos being an average of 2 seconds long and rendered at 25 frames per second, these 10 frames contain data relevant enough to illustrate entire signs. These types of neural networks are unable to sequentially process data hence a Time Distributed layer (TDL) Qiao et al. (2018) is used to consecutively apply the layers of a convolutional network over multiple inputs resulting in the weights being evenly distributed between them. This is computationally cheaper and allows the model to extract features serially over time.

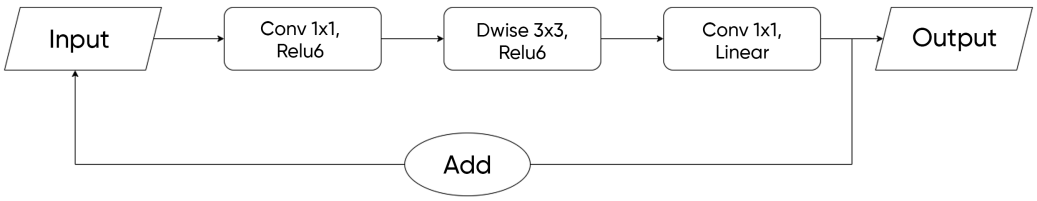


Figure 4.4: MobileNetV2 Architecture

The second part is the CNN where MobileNetV2 Sandler et al. (2018) is implemented because it was lightweight and optimized enough to be employed in real

time scenarios. This version offers enhanced feature learning and understanding of complex features. It also includes a residual connection between its bottleneck layers which allows for accelerated training and finer precision. MobileNetV2 consists of a residual and downsizing block with 3 layers for each. The first layer is known as pointwise convolution consisting of a (1 x 1) convolution combined with ReLU6. It builds new features through calculating linear blends of input channels. The second layer known as depthwise convolution conducts filtering on every input channel. The third is another (1 x 1) convolution layer but linear. The Global Average Pooling layer present in the TDL transforms the 2D vectors inherited from the CNN into 1D vectors to be processed by the RNN.

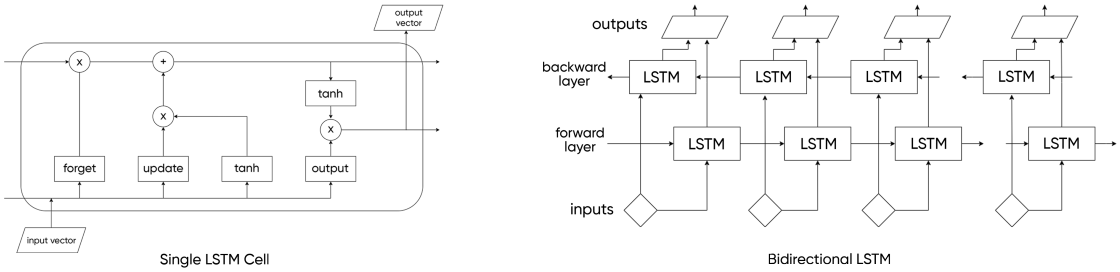


Figure 4.5: Single and Bidirectional LSTM Architecture

The third component was the RNN. Initially, 2 successive GRUs Wang et al. (2022) were employed to extract temporal features due their compact structure and minimal memory requirement. We replaced the GRUs with Bidirectional LSTMs Cui et al. (2018) as they can capture information from both past and future contexts of a series of data and it also performed better upon testing. LSTM Networks Elhagry & Elrayes (2021) as seen in Figure 4.5 use a moving hidden state that enables it to remember continuous data and gates to manage the data. The first gate which consists of a sigmoid layer decides which data to 'forget'. The second layer decides which data to 'update' the cell with and the tanh layer produces potential new values which can be updated later. The final gate generates the cell output and is filtered through a tanh layer. The bidirectional LSTM architecture consists of two different LSTM networks with one processing videos from start to finish and the other from end to beginning. This forward-backward design yields better accuracy on sequential data when compared to regular LSTMs. In this architecture, we stack two layers of Bidirectional LSTMs consecutively and Batch Normalization layers

are inserted between the layers to make training more stable. The outputs from these two layers are concatenated to be used in the fully connected layer.

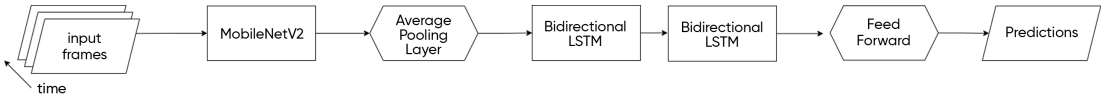


Figure 4.6: C-RNN Architecture

The fully connected layer performs classification based on 1D vector outputs from the RNN and it is made up of a rudimentary feed-forward network made up of dense and dropout layers with a final SoftMax activation layer which produces the scores. The dense layer feeds all outputs it received from the previous layer to all its neurons with each neuron producing singular outputs for the next layer. Dropout is used to reduce over-fitting by adding randomness to the model by randomly setting neurons to zero during training. This randomness encourages the model to learn multiple variations of the data and to be less reliant on any particular features. The final sequential model architecture seen in Figure 4.6, is where the data is processed through all different networks mentioned above in one training phase without the need to train those networks individually.

4.4 Training

Every training is carried out on Google Colaboratory Bisong (2019) due to the high memory demands of the models, with I3D and TGCN being implemented in PyTorch and C-RNN being implemented in Keras Chollet et al. (2015). For I3D and C-RNN we load the videos, pre-trained weights and the splits file into the Colab instance. We used pre-trained ImageNet and Kinetics weights because they are modified to focus on hand motions which enhances feature extraction. For TGCN, we load the key-points instead of videos. The model then reads the data and their classes then divides them for training, validation and testing according to the splits file. This splits file while separating key-points in order of their defined sets also provides data on where the frames start and end for each video.

The number of modifiable training parameters include batch size, epochs, optimizer settings, sample size, hidden size, dense units and the dropout rate. All of the different parameters used for multiple training sessions are listed in the Table 4.1.

The hidden size is the number of hidden units in the GCN layer, the batch size indicates the number of key-points/videos in a batch while the sample number dictates the number of consecutive frames that are randomly picked from each video for training. The use of partial data here forces the model to make predictions from limited data which aids the model in differentiating between different classes and making more accurate predictions by relying on relevant information. This also helps increase the model's performance on new data as they are able to generalize better. As class and samples increased, we modified the hidden size of TGCN and dense units of C-RNN to allocate more neurons to enhance its capacity for modeling complex data and to compensate for the new classes. The dropout rate for C-RNN was set to 0.5 when training with augmented data because it had more data variety. We also changed the batch size, learning rate and hidden size in accordance to avoid issues such as over-fitting or under-fitting.

Table 4.1: Training Parameters

Hyper Parameters	I3D	TGCN	C-RNN
Batch Size	6	64, 256	6
Sample Size	20	50	10
Learning Rate		0.001	
Dropout Rate	0.3	0.3	0.5
Epochs	100	100	100, 200
Loss Function	binary CE with logits	CE	sparse categorical CE
Optimizer		Adam	
Activation Function		SoftMax	

Most batches were trained for 100 epochs with two being trained for 200 epochs due to insufficient analytical results. In each training process, these models learn to map the input frames to their respective class labels. For I3D, horizontal flipping and (224 x 224) cropping are applied randomly frame by frame for each video during this process. Stochastic Gradient Descent (SGD) Liu et al. (2020) was originally used to train I3D but it is not suitable for training the model on smaller datasets, therefore

Adam Optimizer Zhang (2018) was used for training all the models. After the class scores are generated from the fully connected layer, they are compared to true class labels. To calculate the difference between the predicted scores and true class labels, we used different Loss Functions each suited for multi-label classification. Each epoch the model is fed different batches of training data and the weights are updated to reduce the difference between predicted and true class labels. We terminate training and save the best model when no increase in validation accuracy is observed over a number of epochs. Early stopping prevents the model from being over-trained. We test the models to produce evaluation metrics, analytical graphs and confidence matrices which are discussed in the next chapter.

4.5 Web Application

A client-server web application was developed to evaluate the performance of the C-RNN model trained on our dataset. This application can translate sign language hand gestures captured by a webcam and show us the meaning of the gesture on screen. On the client side, a webcam is used to initiate a video stream and 10 frames are cyclically extracted from it. Once the 10-frame buffer is complete, it is sent to the server. The server then uses the trained model to provide a response in accordance with the corresponding gloss. The majority of the workload is handled by the server, which must accurately predict the gloss from a sequence of 10 frames. To implement the server, a Flask framework was utilized. Flask is a lightweight and flexible web framework written in Python, making it ideal for the development of simple yet flexible web applications. Grinberg (2018)

In a traditional client-server setup, communication between the two occurs via the exchange of HTTP calls. However, this approach can be a bottleneck for real-time applications. As HTTP requests are being blocked, the server can only process one request at a time, blocking any other interactions with the client. This could result in a significant delay between the gesture and the server's predicted gloss in the proposed application. To achieve real-time translation, it is necessary to continuously send multiple frames from the client to the server without overwhelming it with a large number of HTTP requests. The solution to this issue is to use a WebSocket. A WebSocket is a protocol that operates over HTTP and provides a

persistent connection between the client and server, enabling bidirectional communication. For this purpose, we use the Flask-SocketIO. The Flask-SocketIO library makes it easy for Flask to enable two-way communication between clients and the server. The client side's sole function is to continuously send groups of 10 frames from the webcam stream through the socket. The prediction is performed by the server using the previously trained model on the frames, which were previously sent as strings and decoded. If the calculated probability surpasses the set threshold, the server returns the predicted gloss.

Chapter 5

Result Analysis

A total of 14 out of many training sessions conducted will be discussed in this Chapter. The majority of the comparison is done on 100 and 200 word sets at 100 epochs. Further training on the 400 and 700 word sets could not be made due to the additional data not being recorded for them. Even with additional samples these sets would be computationally taxing to train on these models. The comparison is based on different performance metrics.

5.1 Performance Metrics

5.1.1 Confusion Matrix

A confusion matrix assists us by providing a comprehensive assessment of a model's strengths and weaknesses. It also provides a visual depiction of how efficiently classifications were made and help identify issues for improvement such as imbalance and misclassification. This assessment is decided through four predictions.

- TP (True Positive) : Number of instances correctly classified as positive.
- TN (True Negative) : Number of instances correctly classified as negative.
- FP (False Positive) : Number of instances wrongly classified as positive.
- FN (False Negative) : Number of instances wrongly classified as negative.

5.1.2 Evaluation Metrics

Accuracy is a common metric for evaluating models. It is the ratio of predictions the models get correctly compared to the total number of predictions. It is defined by the following equation 5.1.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

Precision is a measure of the instances where the true positive predictions made actually turned out to be correct. It is defined by the following equation 5.2.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

Recall is a measure of the true positive predictions that were classified correctly by our models. It is defined by the following equation 5.3.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

F1-Score is an aggregate measure of precision and recall. A relatively high F1-Score indicates that a model has favorable balance between precision and recall. It is defined by the following equation 5.4.

$$F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.4}$$

5.1.3 Functions

A loss function is used to determine the difference between the predicted and true outputs. It measures how well a model is able to make correct predictions. During training, the model parameters are updated based on the gradient of the loss function in order to reduce the error between the outputs. The different functions used below for the models are all used in our research problem which is multi-class classification.

$$L = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{5.5}$$

Binary Cross Entropy with Logits: A loss function where the outputs can only be positive 1 or negative 0. Logits are the non-normalized outputs from the final

layer of the model just before the activation function. It is defined by the following Equation 5.5 where y is the label and p(y) is the predicted probability.

Cross Entropy: A common loss function that computes the cross entropy loss between the input and output logits. It is defined by the following Equation 5.6 where t is the true label and p is the predicted probability.

$$L = -\sum_{i=1}^{n} t_i log(p_i) \tag{5.6}$$

Sparse Categorical Cross Entropy: Commonly used for multi-class classification problems. The difference between categorical and sparse categorical is that the labels are integers rather than one-hot encoded vectors, which makes it more suitable for large amounts of classes. It has the same equation as cross entropy.

5.2 TGCN Performance

All the models were trained on 100 epochs to see how well they perform or falter on the new dataset.

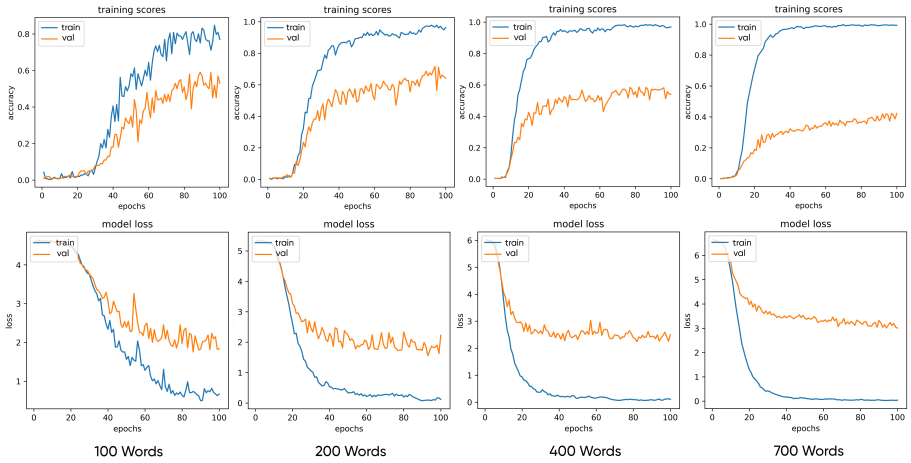


Figure 5.1: Batch A Accuracy and Loss Comparison.

Batch A consists of 3 variations of samples from the original signer which is very bare bones for training. This can be seen in Figure 5.1 by erratic fluctuations in loss and accuracy. It can be seen the model is over-fitting since the training accuracy and validation seems to start off together but then training accuracy keeps increasing as

the validation accuracy slowly plateaus. The training losses are also observed to continually decrease while the validation loss plateaus after a while. As the number of words increases, the gap between them becomes greater which means these are not generalizing well and is guaranteed to perform poorly on unseen data.

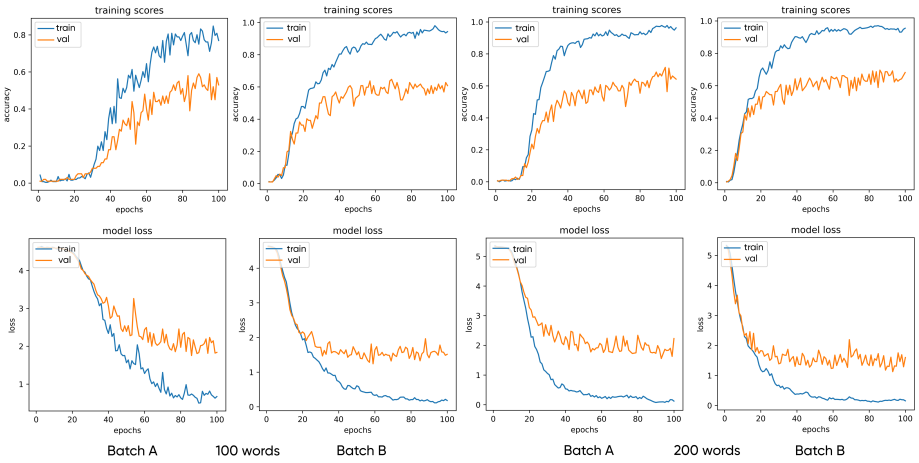


Figure 5.2: Accuracy and Loss Comparison between Batches A and B.

With the inclusion of samples from 3 additional signers in Batch B, the fluctuations in the graph are less erratic. As seen in Figure 5.2 Batch B also performs better in the initial stages but inevitably branches off and starts over-fitting. Six samples for each gloss is not enough.

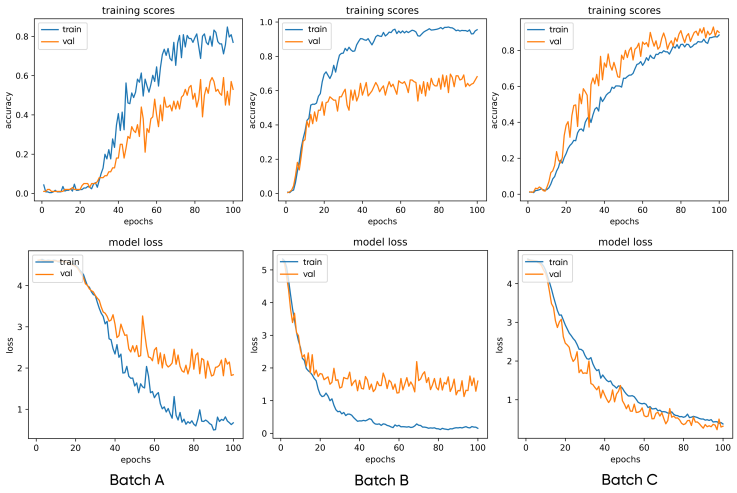


Figure 5.3: Accuracy and Loss Difference for 100 words.

Batch C is the augmented data with 24 different samples for each word. In comparison, it can be seen in Figure 5.2 that the distance between the accuracy and loss curves are minimal but the curves remain somewhat erratic. The validation accuracy is better than training accuracy while the training loss is not decreasing faster than validation loss. This is a sign that the model has not over-fit and is generalizing well.

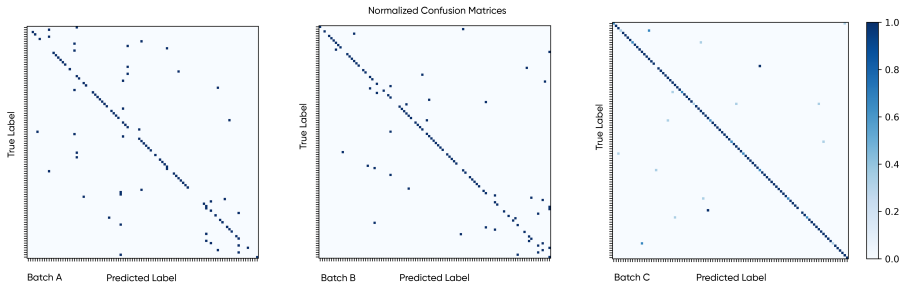


Figure 5.4: Confusion Matrix Comparison for 100 words.

Each row in the confusion matrix represents the true labels and each column represents the predicted labels. From the confusion matrix in Figure 5.4, we see that Batch A has a lot of outliers. These outliers are instances where the model misclassifies signs possibly due to similarity or error. As we move on to Batch B, we can observe that more true positive predictions were made and that there are lesser outliers. For Batch C, the model has made the most correct predictions with the least errors.

Table 5.1: TGCN Performance Overview

Subsets	Batch A			Batch B			Batch C		
Words	T. Acc	V. Acc	Loss	T. Acc	V. Acc	Loss	T. Acc	V. Acc	Loss
100 words	0.88	0.53	1.85	0.92	0.65	1.51	0.90	0.93	0.31
200 words	0.91	0.64	2.22	0.94	0.69	1.60	-	-	-
400 words	0.95	0.54	2.50	-	-	-	-	-	-
700 words	0.96	0.42	3.00	-	-	-	-	-	-

Table 5.2: TGCN Evaluation Overview

Batches		Precision	Recall	F1 Score
Batch A	100 words	0.59	0.58	0.53
	200 words	0.72	0.70	0.65
	400 words	0.59	0.55	0.52
	700 words	0.42	0.42	0.35
Batch B	100 words	0.65	0.62	0.57
	200 words	0.70	0.68	0.63
Batch C	100 words	0.93	0.91	0.92

5.3 I3D Performance

Table 5.3: I3D Evaluation Overview

Batch B	T. Acc	V. Acc	Precision	F1 Score
100 words	0.98	0.57	0.58	0.73
200 words	0.99	0.42	0.40	0.57

Training Batch A would yield extremely over-fit results on Batch A as there were not enough data and without enough samples, these sets were not displayed here. Over-fitting is also seen in Batch B as it has a large disparity in its training and validation accuracies with an uneven F1 Score as seen in Table 5.3. Upon increasing the samples with Batch C, it became apparent it would be both time consuming and computationally expensive to train this model with those subsets. So, how this three dimensional CNN model would perform on the augmented data unfortunately could not be investigated.

5.4 C-RNN Performance

Training Batch B for 100 epochs was not enough as it was not progressing quickly enough to provide meaningful analysis. So, this batch was trained for 200 epochs

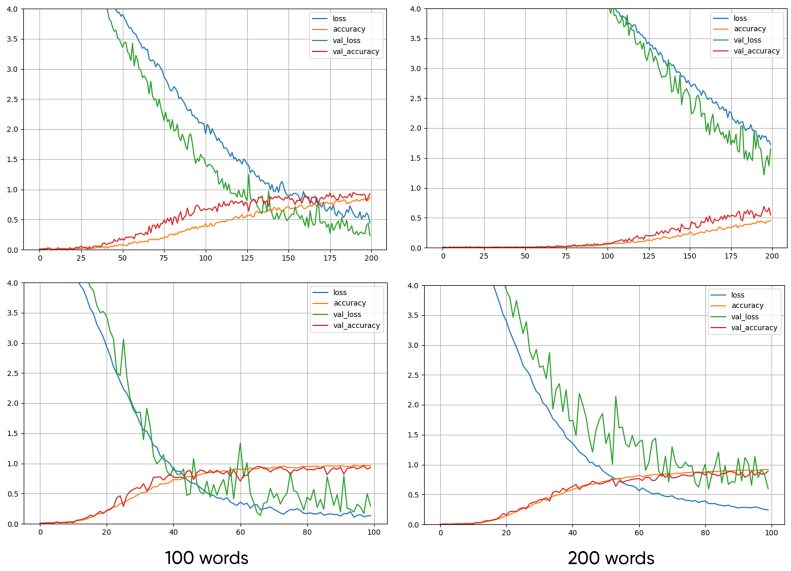


Figure 5.5: Accuracy and Loss Comparison between Batches B and C.

which still was not enough for the 200 words set. The curve starts diverging early after 30 epochs and slowly converging beyond the 200 epoch point as seen in Figure 5.5. Batch C performs better but starts over-fitting much later around 50 epochs which is concerning. The accuracies provided in the table were just before the model started to over-fit for each batch.

Table 5.4: C-RNN Evaluation Overview

Batches		Training	Test	Loss
Batch B	100 words	0.68	0.28	2.09
	200 Words	0.42	0.22	2.79
Batch C	100 Words	0.83	0.87	0.55
	200 Words	0.74	0.76	0.78

5.5 In Comparison

In comparison, it can be seen TGCN performs much better than I3D and C-RNN scoring better in both Batches B and C. The graph model works better because it

does not focus on factors like background or clutter.When compared to similar works done on datasets such as ASL, TGCN achieved an accuracy of 0.55% for 100 words. Compared to our 0.93% accuracy for the same words, we see a 38% increase in performance. CNN-GRU combination attains an accuracy of 0.46% for 100 words and 0.34% for 200 words whereas our modified model which uses a Bidirectional LSTM instead of a GRU has achieved 0.87% and 0.76% accuracies for 100 and 200 words in comparison. This indicates an improvement of 41% and 42% for 100 and 200 words. Since the datasets are different, it is difficult to make a more faithful juxtaposition.

Table 5.5: Model Accuracy Comparison

Metric	Accuracy					
Batch	Batch B			Batch C		
Models	I3D	TGCN	C-RNN	I3D	TGCN	C-RNN
100 words	0.57	0.65	0.28	-	0.93	0.87
200 words	0.42	0.69	0.22	-	-	0.76

5.6 Webcam Testing

We used the model produced from the C-RNN model to carry out a camera test using a webcam to see the detection rates of each word and the model's viability in real time sign recognition. The varying degrees of recognition for some words can be seen in Figure 5.6 below.

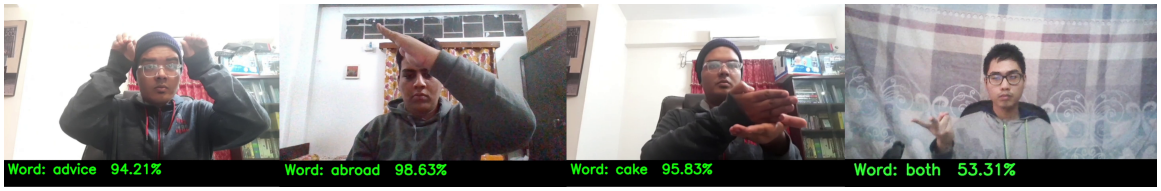
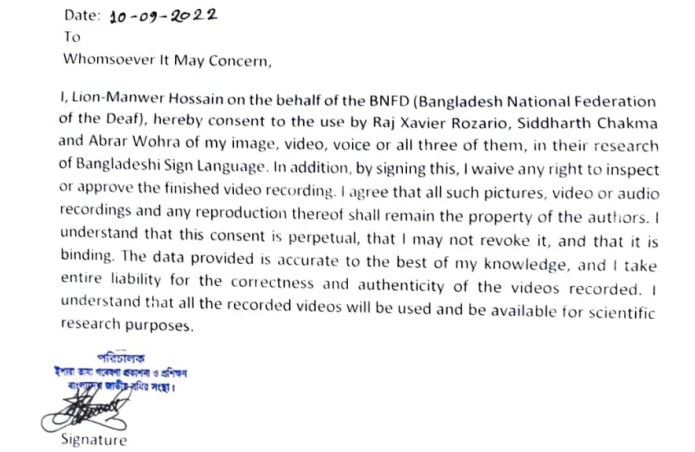


Figure 5.6: Webcam Test

Chapter 6

Conclusions

6.1 Legal and Ethical Issues



Date: 10-09-2022
To
Whomsoever It May Concern,

I, Lion-Manwer Hossain on the behalf of the BNFD (Bangladesh National Federation of the Deaf), hereby consent to the use by Raj Xavier Rozario, Siddharth Chakma and Abrar Wohra of my image, video, voice or all three of them, in their research of Bangladeshi Sign Language. In addition, by signing this, I waive any right to inspect or approve the finished video recording. I agree that all such pictures, video or audio recordings and any reproduction thereof shall remain the property of the authors. I understand that this consent is perpetual, that I may not revoke it, and that it is binding. The data provided is accurate to the best of my knowledge, and I take entire liability for the correctness and authenticity of the videos recorded. I understand that all the recorded videos will be used and be available for scientific research purposes.

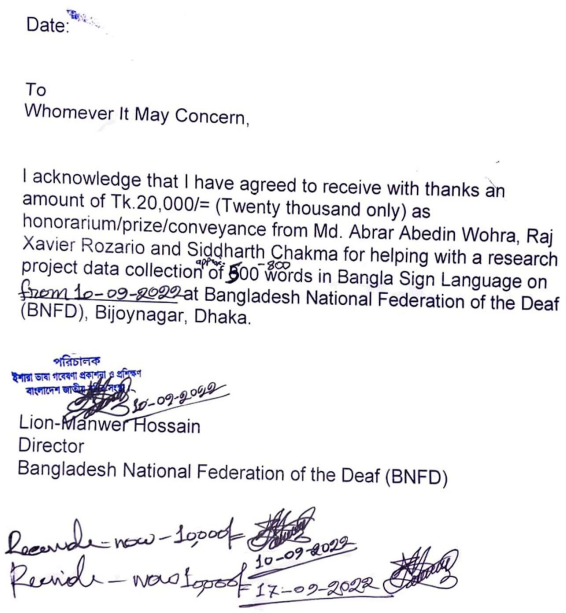
পরিচালক
বাংলাদেশ জাতীয় বধির সংঘ।

Signature

Figure 6.1: Consent Form

Concerns regarding data privacy and transaction receipts came up before we started shooting as we were recording videos of a person's likeness, voice, and image to use for research. We also needed proof of payment. To address this, following the recording consent rules and instructions dictated by of Oregon (2019); of Virginia (2021); Laws (2022), we issued a letter of consent as shown in Figure 6.1 which dictates that he is agreeing to sign over the use of his likeness, voice, image, video, and any further reproduction of those data over to us. We have his consent in binding along with his seal which also holds him liable for the authenticity of the signs and prevents him from revoking his decision. He also acknowledges that the

data will only be used for research purposes. A proof of purchase receipt as seen in Figure 6.2 which clearly states the nature, the purpose, and his involvement in the transaction of Tk. 20,000 had also been drawn. A sign was taken for each day currency changed hands along with an official seal signifying legitimacy from the institution. Therefore, the dataset is not in breach of any confidentiality, privacy or ethical terms. The data we have collected will be privately kept only for research purposes so that it is not misused.



Date:

To
Whomever It May Concern,

I acknowledge that I have agreed to receive with thanks an amount of Tk.20,000/= (Twenty thousand only) as honorarium/prize/conveyance from Md. Abrar Abedin Wohra, Raj Xavier Rozario and Siddharth Chakma for helping with a research project data collection of 500 words in Bangla Sign Language on from 10-09-2022 at Bangladesh National Federation of the Deaf (BNFD), Bijoynagar, Dhaka.

পরিচালক
ইশারা ভাষা গবেষণা প্রকাশনা ও প্রশিক্ষণ
বাংলাদেশ জাতীয় [illegible]
10-09-2022

Lion-Manwer Hossain
Director
Bangladesh National Federation of the Deaf (BNFD)

Receive = now - 10,000/= 10-09-2022
Remain - now 10,000/= 17-09-2022

Figure 6.2: Payment Receipt

6.2 Social Impact

With further improvements, the models worked on here can be implemented to contribute to rising social issues here in Bangladesh. The primary of them is the reason we started this project, communication breakdown. Disabled people can only communicate with people who know sign language which is a rare skill here. Otherwise, they would have to hire an interpreter to translate for them which can be costly over time and most cannot bear that cost. This communication gap prevents them from getting opportunities at work and even at life. There is a

growing demand for more sign language education and researchers have started using machine learning to meet those demands. The implementation of our work into a mobile or web application can remedy this gap in communication by allowing people to translate what the impaired are saying while also learning the language in the process. By making it an application, it can be available to anyone who owns a phone while also cutting the need of an interpreter. Our dataset was recorded with the most recent updates to the signs and can be further enhanced with more classes and samples to be a step closer to understanding the entirety of the language.

6.3 Conclusion

This project aimed to create a deep learning based system to recognize signs from a custom video dataset. We collected a unique dataset consisting of 700 bangla sign words which probably has never been done before. With different subsets of classes, we trained on three different types of models which were compared and analyzed for the best suited and most optimal model for efficient sign recognition. We found that TGCN achieved the best scores with 0.93 accuracy with the lowest loss score of 0.31. Over-fitting was a significant challenge which we tried to address through the scaling of data but with more signer variety and sample numbers this issue could be eliminated. A web application was also developed for the purpose of demonstration. We hope to further develop the dataset so that it could yield better results when used in real time detection and future research in word level BdSL.

References

Araujo, A., Norris, W. D., & Sim, J. (2019). Computing receptive fields of convolutional neural networks. *Distill*.

Bisong, E. (2019). Google colaboratory. In *Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners* (pp. 59–64). Berkeley, CA: Apress. Retrieved from https://doi.org/10.1007/978-1-4842-4470-8_7 doi: 10.1007/978-1-4842-4470-8_7

Boháček, M., & Hrúz, M. (2022). *Sign pose-based transformer for word-level sign language recognition.* Retrieved from https://developer.apple.com/documentation/

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., & Sheikh, Y. (2018, 12). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. Retrieved from http://arxiv.org/abs/1812.08008

Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2016). *Realtime multi-person 2d pose estimation using part affinity fields.* arXiv. Retrieved from https://arxiv.org/abs/1611.08050 doi: 10.48550/ARXIV.1611.08050

Carreira, J., & Zisserman, A. (2017, 5). Quo vadis, action recognition? a new model and the kinetics dataset. Retrieved from http://arxiv.org/abs/1705.07750

Cauli, N., & Recupero, D. R. (2022, 3). Survey on videos data augmentation for deep learning models. *Future Internet*, *14*. doi: 10.3390/fi14030093

Chollet, F., et al. (2015). *Keras.* https://github.com/fchollet/keras. GitHub.

Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). *Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction.* arXiv. Retrieved from https://arxiv.org/abs/1801.02143 doi: 10.48550/ARXIV.1801.02143

de Amorim, C. C., Macêdo, D., & Zanchettin, C. (2019, 1). Spatial-temporal graph convolutional networks for sign language recognition. Retrieved from http://arxiv.org/abs/1901.11164http://dx.doi.org/10.1007/978-3-030-30493-5_59 doi: 10.1007/978-3-030-30493-5_59

Elhagry, A., & Elrayes, R. G. (2021). *Egyptian sign language recognition using cnn and lstm.* arXiv. Retrieved from https://arxiv.org/abs/2107.13647 doi: 10.48550/ARXIV.2107.13647

Fahad, A., Hassan, H., Abdalla, S., & Hassan, H. (2020, 05). Deep learning-based deaf mute gesture translation system. *International Journal of Science and Research (IJSR)*, *9*, 288-292. doi: 10.21275/SR20503031800

Grinberg, M. (2018). *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

Hegde, S., & Gangisetty, S. (2021, 4). Pig-net: Inception based deep learning architecture for 3d point cloud segmentation. *Computers and Graphics (Pergamon)*, *95*, 13-22. doi: 10.1016/j.cag.2021.01.004

Hosain, A. A., Santhalingam, S., Pathak, P., Rangwala, H., & Košecká, J. (2021). *Hand pose guided 3d pooling for word-level sign language recognition.*

Hossen, M. A., Govindaiah, A., Sultana, S., & Bhuiyan, A. (2019, 2). Bengali sign language recognition using deep convolutional neural network. In (p. 369-373). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICIEV.2018.8640962

Jahan, N. (2019). *The disabled need more than an allowance.* https://www.thedailystar.net/star-weekend/news/the-disabled-need-more-allowance-1759756. The Daily Star.

Jiang, S., Sun, B., Wang, L., Bai, Y., Li, K., & Fu, Y. (2021, 10). Sign language recognition via skeleton-aware multi-model ensemble. Retrieved from http://arxiv.org/abs/2110.06161

Karim, H. R. K. . A. (1994). *Bangla sign language dictionary.* Bangladesh National Federation of the Deaf. Retrieved from https://books.google.com.bd/books?id=oa_0HAAACAAJ

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., ... Zisserman, A. (2017, 5). The kinetics human action video dataset. Retrieved from http://arxiv.org/abs/1705.06950

Kipf, T. N., & Welling, M. (2016, 9). Semi-supervised classification with graph convolutional networks. Retrieved from http://arxiv.org/abs/1609.02907

Ko, S. K., Kim, C. J., Jung, H., & Cho, C. (2019, 7). Neural sign language translation based on human keypoint estimation. *Applied Sciences (Switzerland)*, *9*. doi: 10.3390/app9132683

Kornblith, S., Chen, T., Lee, H., & Norouzi, M. (2020). Why do better loss functions lead to less transferable features? In *Neural information processing systems.*

kuang Chiu, H., Adeli, E., Wang, B., Huang, D.-A., & Niebles, J. C. (2018, 10). Action-agnostic human pose forecasting. Retrieved from http://arxiv.org/abs/1810.09676

Laws, R. (2022, Sep). *Bangladesh audio and video recording laws.* Retrieved from https://recordinglaw.com/bangladesh-recording-laws/#:~:text=Bangladesh%20is%20a%20two%2Dparty,or%20phone%20call%20before%20recording.

Li, D., Opazo, C. R., Yu, X., & Li, H. (2019, 10). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. Retrieved from http://arxiv.org/abs/1910.11006

Li, D., Xu, C., Yu, X., Zhang, K., Swift, B., Suominen, H., & Li, H. (2020, 10). Tspnet: Hierarchical feature learning via temporal semantic pyramid for sign language translation. Retrieved from http://arxiv.org/abs/2010.05468

Li, G., Zhang, M., Li, J., Lv, F., & Tong, G. (2020, 08). Efficient densely connected convolutional neural networks. *Pattern Recognition*, *109*, 107610. doi: 10.1016/j.patcog.2020.107610

Lipi, K. A., Adrita, S. F. K., Tunny, Z. F., Munna, A. H., & Kabir, A. (2022). Static-gesture word recognition in bangla sign language using convolutional neural network. *Telkomnika (Telecommunication Computing Electronics and Control)*, *20*, 1109-1116. doi: 10.12928/TELKOMNIKA.v20i5.24096

Liu, Y., Gao, Y., & Yin, W. (2020). *An improved analysis of stochastic gradient descent with momentum.* arXiv. Retrieved from https://arxiv.org/abs/2007.07989 doi: 10.48550/ARXIV.2007.07989

Maruyama, M., Ghose, S., Inoue, K., Roy, P. P., Iwamura, M., & Yoshioka, M. (2021, 6). Word-level sign language recognition with multi-stream neural networks focusing on local regions. Retrieved from http://arxiv.org/abs/2106.15989

Meena, G., Mohbey, K. K., Indian, A., & Kumar, S. (2022, jan). Sentiment analysis from images using vgg19 based transfer learning approach. *Procedia Comput. Sci.*, *204*(C), 411–418. Retrieved from https://doi.org/10.1016/j.procs.2022.08.050 doi: 10.1016/j.procs.2022.08.050

Miah, A. S. M., Shin, J., Hasan, M. A. M., & Rahim, M. A. (2022, 4). Bensignnet: Bengali sign language alphabet recognition using concatenated segmentation and convolutional neural network. *Applied Sciences (Switzerland)*, *12*. doi: 10.3390/app12083933

Noori, F. M., Wallace, B., Uddin, M. Z., & Torresen, J. (2019). A robust human activity recognition approach using openpose, motion features, and deep recurrent neural network. In *Image analysis: 21st scandinavian conference, scia 2019, norrköping, sweden, june 11–13, 2019, proceedings* (p. 299–310). Berlin, Heidelberg: Springer-Verlag. Retrieved from https://doi.org/10.1007/978-3-030-20205-7_25 doi: 10.1007/978-3-030-20205-7_25

of Oregon, U. (2019). *Ovpri: Guidance on audio, video, and/or photography.* Retrieved from https://research.uoregon.edu/manage/research-integrity-compliance/human-subjects-research/audio-recording-video-recording-andor-photography

of Virginia, U. (2021). *Informed consent for recording devices.* Retrieved from https://research.virginia.edu/irb-sbs/informed-consent-recording-devices

Podder, K. K., Chowdhury, M. E., Tahir, A. M., Mahbub, Z. B., Khandakar, A., Hossain, M. S., & Kadir, M. A. (2022, 1). Bangla sign language (bdsl) alphabets and numerals classification using a deep learning model. *Sensors*, *22*. doi: 10.3390/s22020574

Qiao, H., Wang, T., Wang, P., Qiao, S., & Zhang, L. (2018). A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series. *Sensors*, *18*(9). Retrieved from https://www.mdpi.com/1424-8220/18/9/2932 doi: 10.3390/s18092932

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

Rahman, M. M., Mdrafi, R., Gurbuz, A. C., Malaia, E., Crawford, C., Griffin, D., & Gurbuz, S. Z. (2021). *Word-level sign language recognition using linguistic adaptation of 77 ghz fmcw radar data.*

Sadik, F., Subah, M. R., Dastider, A. G., Moon, S. A., Ahbab, S. S., & Fattah, S. A. (2019, 11). Bangla sign language recognition with skin segmentation and binary masking. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/WIECON-ECE48653.2019.9019931

Sakib, S. N. (2019). *Hearing-impaired demand sign language institute in bangladesh.* https://www.aa.com.tr/en/asia-pacific/hearing-impaired-demand-sign-language-institute-in-bangladesh/2371702. Anadolu Ajansi.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018, 06). Mobilenetv2: Inverted residuals and linear bottlenecks. In (p. 4510-4520). doi: 10.1109/CVPR.2018.00474

Sharma, A., Singh, A., Choudhury, T., & Sarkar, T. (2021, 04). Image classification using imagenet classifiers in environments with limited data. doi: 10.21203/rs.3.rs-428416/v1

She, D., Lai, Y.-K., Yi, G., & Xu, K. (2021). Hierarchical layout-aware graph convolutional network for unified aesthetics assessment. In *2021 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 8471-8480). doi: 10.1109/CVPR46437.2021.00837

Shi, L., Zhang, Y., Cheng, J., & Lu, H. (2018). Two-stream adaptive graph convolutional networks for skeleton-based action recognition. Retrieved from https://arxiv.org/abs/1805.07694 doi: 10.48550/ARXIV.1805.07694

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

Tajmim, J. I. . T. (2022). *Number of persons with disabilities twice than previously assessed.* https://www.tbsnews.net/bangladesh/health/number-persons-disabilities-twice-previously-assessed-440522. The Business Standard.

Talukder, D., & Jahara, F. (2020, 12). Real-time bangla sign language detection with sentence and speech generation. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICCIT51783.2020.9392693

Tian, L., Cheng, X., Honda, M., & Ikenaga, T. (2022). Multi-view 3d human pose reconstruction based on spatial confidence point group for jump analysis in figure skating. *Complex and Intelligent Systems*. doi: 10.1007/s40747-022-00837-z

Urmee, P. P., Mashud, M. A. A., Akter, J., Jameel, A. S. M. M., & Islam, S. (2019, 11). Real-time bangla sign language detection using xception model with augmented dataset. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/WIECON-ECE48653.2019.9019934

van den Brand, J., Lee, Y.-T., Nanongkai, D., Peng, R., Saranurak, T., Sidford, A., ... Wang, D. (2020, 9). Bipartite matching in nearly-linear time on moderately dense graphs. Retrieved from http://arxiv.org/abs/2009.01802

Vo, A. H., Pham, V. H., & Nguyen, B. T. (2019, 8). Deep learning for vietnamese sign language recognition in video sequence. *International Journal of Machine Learning and Computing*, *9*, 440-445. doi: 10.18178/ijmlc.2019.9.4.823

Wang, J., Zou, Y., & Lim, S.-J. (2022, 11). *An improved time feedforward connections recurrent neural networks.*

Wu, L., Sun, P., Hong, R., Fu, Y., Wang, X., & Wang, M. (2018, 11). Socialgcn: An efficient graph convolutional network based model for social recommendation. Retrieved from http://arxiv.org/abs/1811.02815

Yin, Q., Zhang, R., & Shao, X. (2019, 01). Cnn and rnn mixed model for image classification. *MATEC Web of Conferences*, *277*, 02001. doi: 10.1051/matecconf/201927702001

Zhang, Z. (2018). Improved adam optimizer for deep neural networks. In *2018 ieee/acm 26th international symposium on quality of service (iwqos)* (p. 1-2). doi: 10.1109/IWQoS.2018.8624183

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., ... Li, H. (2018, 11). T-gcn: A temporal graph convolutionalnetwork for traffic prediction. Retrieved from http://arxiv.org/abs/1811.05320http://dx.doi.org/10.1109/TITS.2019.2935152 doi: 10.1109/TITS.2019.2935152

Zhou, C., & Li, W. (2019). Pose comparison based on part affinity fields*. *2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP)*, 519-523.