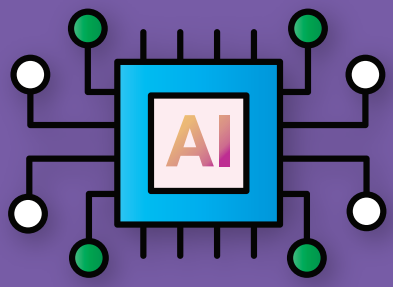
genweb2

We commit to what we can deliver,
We deliver what we commit



January 26, 2022

INCEPTION REPORT

Development of Software for Disable People (SD-17)

Enhancement of Bangla Language in ICT through Research & Development (1st revised)

Version 1.0.0

Submitted To -

PROJECT DIRECTOR

Enhancement of Bangla Language in ICT through Research & Development Project (1st revised)
EBLICT Project Office
ICT Tower (8th floor), Agargaon,
Dhaka-1207, Bangladesh

TK Bhaban (11th Floor), 13 Kazi Nazrul Islam Avenue, Kawran Bazar Dhaka-1215
Phone:+88 02 55013600, 55013601, 55013602, Fax: +88 02 55013602

Date: 26-01-2022 Ref: GW2/EBLICT/IR/01/01

To
Md. Mahbub Karim
Project Director
Enhancement of Bangla Language in ICT through Research & Development (1st Revised) Project
Bangladesh Computer Council
Information & Communication Technology Division
Ministry of Posts, Telecommunications and Information Teclrnology
ICT Tower, Agargoan
Shere Bangla Nagar, Dhaka-1207.

Subject: Submission of Inception Report (V-1.0.0).

Dear Sir,

With due respect and humble submission, this is to state that in accordance with the agreement (Contract Ref: 56.01.0000.029.32.039.17.648) between Genweb2 Limited and your esteem project of we hereby submitting the inception report (V-1.0.0) for the assignment of "Development of software to disable people (SD-17)" under EBLICT project.

Please accept and acknowledge the receipt of the inception report and kindly let us know if any further assistance is required.

Sincerely yours,

_____

Md. Obydul Islam
General Manager- Operations
Genweb2 Limited
Cell: +880 1817184581
Email: islam.obydul@genweb2.com

Project Information	
Project Name	"Enhancement of Bangla Language in ICT through Research & Development (1st revised)" Project
Consulting Service Title	Development of Software for Disable People (SD 17)
Consulting Service Awarded To	Genweb2 Limited
Contract Reference Number	56.01.0000.029.32.039.17.648
Funded By	Government of the People's Republic of Bangladesh Information & Communication Technology Division Bangladesh Computer Council

Deliverable Information	
Order of Delivery	1
Name of Deliverable	Inception Report
Deliverable Type	Report
Due Date	31.01.2022
Submission Date	26.01.2022
Version	1.0.0
Prepared & Submitted By	Genweb2 Limited
Authors	Hossain All Nahhyan, Jobaidul Alam Jamali, Shahidul Islam
Reviewers	Rafiul Alam, Md. Shahidul Haque

Version control			
Version Number	Issued Date	Revision Date	Description of Change
V1.0.0	10.06.2019	N/A	N/A

Table of Content

Table of Figures

1. Executive Summary

Hearing impairment is the second commonest form of disability in Bangladesh and is causing economic, social, educational, and vocational problems. Children are the worst affected section of the community in this regard. Many factors including genetic factors, childhood diseases including diseases of the mother during pregnancy and exposure to noise above the threshold level over an extended period are responsible for causing hearing impairment. Deafness produces substantial social and economic costs throughout the world because of its effect on child health development and education.

Development of Software for Disable People (SD-17) brings us the opportunity to upstream deaf community who has never been considered to contribute to the national development of the country. The proposed solution will bring bidirectional communication between deaf and normal people and also will introduce a new mechanism of education system in blind or low vision community as well using the braille converter.

The major two components of SD-17 is SLR and T2SP for Bangla language that will help to establish an unprecedented communication channel between deaf community and normal people. The core functionalities of two components are to convert Bangla sign language into standard Bangla Unicode text and translate Bangla text into sign/gesture through puppets respectively. Very few researches have been conducted for the Bengali Sign Language recognition and translations. With the latest advancement of different AI algorithms - no initiatives have been taken probably because Bangla is currently an under-resource language and preparing a very large/massive dataset on Bangla corpus is a very big challenge.

Most recently, technological advancement (CPU and GPU) and massive improvements in AI/Deep learning algorithm have enabled the door to achieve near close prediction in machine and deep learning field. A prominently right time to take the challenge to prepare a very large/massive dataset on Bangla corpus, trained sufficiently with latest state-of-art deep learning is expected to provide significant recognition accuracy for Bangla Sign (considering we have a set of standard Bangla sign language) and translation. Similarly, converting text to sign puppet or avatar has become a noticeably a remarkable sector of modern researches and success is atop.

Different companies and organizations doing rapid researches and development on this field, noticeably some of them are:

- DGS Korpus [1] -The aim is to collect sign language texts from deaf community and to present parts of them as a Public DGS Corpus. A 15 year's running project targets a corpus-based electronic DGS – German dictionary.
- SignAll [2] has the largest database of ASL vocabulary in the world. SignAll has built a strong partnership with Gallaudet University (GU) for continuous research and development with latest advancements.
- KinTrans [3] - Uses AI to learn and process the body movements of sign language.
- Microsoft [4] - Their system is still a research prototype but advancing rapidly.

More advanced algorithm and frameworks can help us to convert Bangla Sign Language to Text more accurately. We can follow the approaches that big companies are investing on, like

Detectron which is a Facebook AI Research's (FAIR) software system that implements state-of-the-art object detection algorithms from motion images, including Mask R-CNN. DensePose and PoseNet are far cleaner algorithms for object detection to the details that can also be used as well.

Sign to Text and text to puppet are assistive applications, will be designed for use by disabled people. These two modules will be very helpful for struggling deaf and speech disable community to exchange views and ideas with normal people. These applications will definitely enable new door of communication for the deaf community, mentioned a few below out of many:

- General conversation between deaf community and normal people.
- Establish communication in educational sector.
- Establish communication in professional career opportunity.
- Establish communication in court and legal activities where deaf community can't explain as victims.
- Take part in national development stream with the help of this modern technical assistance.

To support these two major components a database will be constructed containing all Bangla alphabets, numerals, and words used in BdSL by mapping them with their corresponding BdSL gestures (aligned dataset) so that respective stakeholders can get benefited as and when needed.

For enhanced education system for blind and low vision community a Bangla Unicode text to Bangla braille conversation tool with editor will be developed that will be equipped with math translation, generate emboss image on braille, language switching option and finally print capability in major commercial Braille embossers.

A cloud-based data collection & processing interface will be built enabling crowdsourcing and validation that will help to provide dataset management, labeling and validation tool with necessary web interface as the workgroup can contribute to enrich the dataset with inbuilt administrative activities.

Genweb2 Ltd. has a professional expertise and experience for several years in the relevant government projects development. A prominent and contemporary AI expert team has been deployed for this particular problem with the intention to achieve close to match success. Genweb2 strongly believes that the team will be able to make the assignment successful and bring dynamicity and efficiency that the Bangladesh Computer Council is looking for.

1.1 Understanding of the Assignment

The core purpose of this project is to develop a set of modules as deliverables to support the disable people and removing obstacle of communication. The first module comes as a converter of Bangla sign language. The core functionality of BdSLR engine is to recognize Bangla sign (from recorded or continuous video stream) and convert the BDSL into Bangla standard Unicode text. The crux parts of this module are sign converters that will do the followings:

- Recognize Bangla sign language from recorded and real-time motion images.
- Output the standard Bangla Unicode text from the recognized Bangla sign.
- Convert Sign-Bangla to standard Bangla.

The second module comes as a converter of Bangla Text to Bangla Standard Sign generator through puppet (T2SP). The crux part of this module is a text to gesture translation system that will do the followings:

- Recognize standard Bangla text inputted by users.
- Output sign/gesture through puppets (male/female).

The third module comes as a Braille converter with editor (B2BC). The crux part of this module is a text to Bangla braille conversion (B2BC) with an editor tool that will do the followings:

- Recognize Bangla Unicode text inputted by users.
- Output Braille Code (Braille symbols that will be formed within units of space known as braille cells in the braille editor).

Apart from these three major conversion and translation modules we will consider Sign Language Communicator for Chat application. In addition, separate applications will be built for hand-handled devices such as Android and iOS using the SLR and T2SP converter. The user interface for the mobile applications will be user friendly for Speech & Hearing Disable community. To enhance the Bangla corpus a crowdsourcing an application will be built so that acceptable sign can received from public community and train our model accordingly.

Finally, developed modules will expose/consume API handshaking with external applications without complexity. For example, the SLR module will handshake with external systems like TTS. And also, SLR module will expose APIs for Corpus/Data transfer, API for third party service (with token system), and API for SD19.

2 Project Components and Modules

2.1 Sign Language Recognition and Conversion (SLR)

A core system/solution need to be implemented which will be able to recognize Bangla Sign Language (BDSL code set) and convert that sign into Bangla Unicode text. With this system implemented, real-time motion recorded video/image of sign language with gestures will be captured as input, Bangla Unicode will be the output.

2.2 Bangla Text to Sign generator through 3D-model/ Puppet (T2SP)

The proposed module will take Bangla text as inputs and produces corresponding sign language, represented by a 3D animated agent (puppet/avatar) in real-time for the deaf community so that they can understand normal peoples conversation. Output animation would be rendered using a computer-generated model.

2.3 Sign Language Communicator for Chat (SLC)

This chat application will come up with conventional and established features and functionalities of popular chat application along with sign language chat. Following are the functionalities of the Bangla sign language enabled chat application:

- Register, Create and manage user profiles.
- Create, Manage one or multiple groups.
- Bidirectional Instant messaging with other recipients (Sign to text, text to Puppet).
- Reply on particular message (based on message thread)
- They will also be able to invite people to join in chat groups etc.

2.4 Mobile APP for SLR (Android & IOS)

Developing Bangla sign language recognition methods for mobile applications has challenges like need for light weight method with less CPU and memory utilization. As a part of this project, a deep learning-based web enabled Sign Language Recognition (SLR) System will be developed where we will get all necessary API suite for sign language to text translation. The application will captures the video stream using device camera and submit it for translation using API of SLR server-end and receive corresponding texts, this is a server based processing. By using those API our proposed mobile application will enable commutation through chat for speech & hearing disable community. We will also try to give an in memory processing for better performance, if satisfactory results come in we will accept that. The application allows inclusive communication of speech & hearing disable community with other actors of the society.

Features and Functionalities of Mobile APP

- **Bangla Sign Language Translator:** The translator app will be able to recognize Bangla Sign Language (BDSL code set) and convert that sing into Bangla Unicode text. With this application implemented, real-time motion recorded video/image of Bangla sign language with gestures will be captured as input, Bangla Unicode text will be the output on the other side.

- **Bangla Text to Sign Generator:** This part of the translator app will be able to convert Bangla text to Bangla sign language (using puppet/avatar). More particularly, the translator will be able to recognize Bangla text and display Bangla sign language via an animated puppet. So this will help to make understand normal people's conversation to the deaf community very clearly which is now quiet impossible.
- **Bangla Sign Language communicator for Chat:** The chat application will enable instant messaging feature between deaf community and normal people and vice a versa, which will offer real-time messaging conversation over the Internet. User may perform Bangla sign language and application will recognize corresponding Bangla Unicode text. Those text can be sent as messages to the other recipient of the chat application.

2.5 A cloud-based data collection & processing interface

Data collection is a major bottleneck in machine learning and an active research topic in multiple communities. There are largely two reasons data collection has recently become a critical issue. First, as machine learning is becoming more widely used, we are seeing new applications that do not necessarily have enough labeled data. Second, unlike traditional machine learning where feature engineering is the bottleneck, deep learning techniques automatically generate features, but instead require large amounts of labeled data. Interestingly, recent research in data collection comes not only from the machine learning, natural language, and computer vision communities, but also from the data management community due to the importance of handling large amounts of data.

2.6 BdSL Enabled Web Interface

Apart from mobile application, a web interface will be developed for sign language recognition and translation. The entire web application will be comprised with the following components:

- SLR Module
- T2SP Module
- API Module
- Chat Module
- Crowd sourcing and Validation Module

2.7 Braille converter with editor

Rewriting of books into Braille code is accomplished by certified Braille experts. But this process takes time and cost of translation is also higher. For this reason, books written in Braille code are insufficient. This is a potential constraint in the education for visually impaired. This constraint can be overcome by an automated translation system which will be able to convert Bangla Unicode text to Bangla Braille code with all grammatical considerations and standards

without any error. In order to improve this scenario, a converter for facilitating Bangla Unicode text to Bangla Braille.

2.8 Office Extension of B2BC

Office ADD-IN

Office Add-ins platform used to build solutions that extend Office applications and interact with content in Office documents. With Office Add-ins, you can use familiar web technologies such as HTML, CSS, and JavaScript to build a solution that can run in Office across multiple platforms including Windows, Mac, iPad, and in a web browser. In Office Add-ins platform we have option to build Add-in to extend default features for all the components of Office suite like Word, Excel, PowerPoint, Outlook, Visio etc. In this project we are assuming that we need to develop a MS Word extension of B2BC converter.

With Word add-ins, we can use familiar web technologies such as HTML, CSS, and JavaScript to build a solution that extends the functionality of Word and can run in Word across multiple platforms including Windows, Mac, iPad, and in a web browser. Word add-ins are one of the many development options that you have on the Office Add-ins platform. We will use add-in commands to enable Braille printing features in MS Word.

2.9 API for SLR, T2SP & B2BC

API should exposed so that the system should be easily integrated with external applications.

Technically, REST endpoint need to be exposed so that other system can easily handshake and get the necessary services. With adequate authentication and authorization (Token Based) API will be exposed from this BdSLR application (SLR, T2SP and Braille Converter).

2.10 Project Deliverables

Task no	Task	Desired size/ performance
1	1. Inception Report.	• Detailed report on the understanding of the concept, and work plan, Design Documents, SRS with sample dataset.
2	1. Gold Standard Aligned dataset (Video to Text) - 5%. 2. Developing a cloud-based sign dataset collection & processing interface: Version 1.0. 3. Basic web version for SLR 1.0 (Sign Recognition Module+ Sign-Text to Bangla Text Module).	• Dataset A: At least 25K Sentences with 100+ minute corresponding labeled video. • Dataset B: At least 25K gloss with 100+ minute corresponding labeled video. • Sign Gloss from Recorded and Real-time Video 25%.
3	1. Web version for SLR 1.1. 2. Bangla Unicode to Braille converter (B2BC) 1.0. 3. Braille Books Printed using converter. 4. Basic web version for T2SP 1.0.	• Sign Gloss from Recorded and Real-time Video 35%. • Sentence from Recorded and Real-time Video 30%. • 5 books (50 copies of each book). Each book containing 200-250 both side braille printed sheets.
4	1. Web version for SLR version 1.2. 2. Basic web version of Sign Language Communicator Version 1.0 (integrated with TTS and STT). 3. Basic Mobile App (Android & iOS) version 1.0. 4. Unicode to Braille converter with editor (B2BC) version 1.1. 5. Braille Books Printed using converter 6. Basic web version for T2SP 1.1. 7. Gold Standard Aligned dataset (Video to Text) 8. Gold Standard Aligned Dataset (Text to Gloss)	• Sign Gloss from Recorded and Real-time Video 55%. • Sentence from Recorded and Real-time Video 50%. • Dialogue from Recorded and Real-time Video 50%. • Dataset A: At least 500K Sentences with corresponding labeled video. • UAT score 4.0 out of 5 for Braille. • 20 books (50 copies of each book). Each book containing 200-250 both side braille printed sheets.
5	1. SLR version 1.3. 2. Sign Language Communicator for Chat 1.1. 3. Mobile App (Android & iOS) version 1.1. 4. Basic web version for T2SP 1.2.	• Sign Gloss from Recorded and Real-time Video 70%. • Sentence from Recorded and Real-time Video 65%. • Dialogue from Recorded and Real-time Video 60%.

6	1. Web version of total solution (SLR+ SLC+ interface) Version 2.0. 2. Mobile App (Android & iOS) version 2.0. 3. Basic web version for T2SP 2.0. 4. Android and IOS for T2SP. 5. A cloud-based data collection & processing interface (with Crowd sourcing & Validation Module): Version 1.1 Office extension of two solution (SLR and B2BC). 6. APIs: 6. A: API for developer of Recognition engine. 6. B: API for Corpus/Data transfer. 6. C: API for third party service. 7. Braille Books Printed using converter. 8. Integrated Management interface.	• Sign Gloss from Recorded and Real-time Video 75%. • Sentence from Recorded and Real-time Video 70%. • Dialogue from Recorded and Real-time Video 65%. • UAT for SLR 4 out of 5. • 75 books (50 copies of each book). Each book containing 200-250 both-side braille printed sheets.
7	1. Fine-tuning of the system/s after getting users' feedback and pass through UAT. 2. Web version to total solution 3.0 (Final Version). 3. Other tools/resource, mass-level testing after public release and reception of all documentations.	• Sentence-Video dataset: At least 300 hours. • Gloss-Video dataset: At least 100 hours • Sign Gloss from Recorded and Real-time Video 75%. • Sentence from Recorded and Real-time Video 70%. • Dialogue from Recorded and Real-time Video 65%. • UAT score 4.5 out of 5.

2.11 Project Duration

Task no	Task	Time of delivery (month)
1	1. Inception Report.	2nd Month :: January 2022
2	1. Gold Standard Aligned dataset (Video to Text) - 5%. 2. Developing a cloud-based sign dataset collection & processing interface: Version 1.0. 3. Basic web version for SLR 1.0 (Sign Recognition Module+ Sign-Text to Bangla Text Module).	4th Month :: Mar 2022
3	1. Web version for SLR 1.1. 2. Bangla Unicode to Braille converter (B2BC) 1.0. 3. Braille Books Printed using converter. 4. Basic web version for T2SP 1.0.	8th Month :: Jul 2022
4	1. Web version for SLR version 1.2. 2. Basic web version of Sign Language Communicator Version 1.0 (integrated with TTS and STT). 3. Basic Mobile App (Android & iOS) version 1.0. 4. Unicode to Braille converter with editor (B2BC) version 1.1. 5. Braille Books Printed using converter 6. Basic web version for T2SP 1.1. 7. Gold Standard Aligned dataset (Video to Text) 8. Gold Standard Aligned Dataset (Text to Gloss)	12th Month :: November 2022
5	1. SLR version 1.3. 2. Sign Language Communicator for Chat 1.1. 3. Mobile App (Android & iOS) version 1.1. 4. Basic web version for T2SP 1.2.	14th Month :: January-2023
6	1. Web version of total solution (SLR+ SLC+ interface) Version 2.0. 2. Mobile App (Android & iOS) version 2.0. 3. Basic web version for T2SP 2.0. 4. Android and IOS for T2SP. 5. A cloud-based data collection & processing interface (with Crowd sourcing & Validation Module): Version 1.1 Office extension of two solution (SLR and B2BC). 6. APIs: 6. A: API for developer of Recognition engine. 6. B: API for Corpus/Data transfer. 6. C: API for third party service. 7. Braille Books Printed using converter. 8. Integrated Management interface.	16th Month :: March 2023
7	1. Fine-tuning of the system/s after getting users' feedback and pass through UAT. 2. Web version to total solution 3.0 (Final Version).	18th Month :: July 2023

	3. Other tools/resource, mass-level testing after public release and reception of all documentations.	

3 Technical Approaches & Methodologies

3.1 Plan to Attack the Problem

Two most critical components are BdSLR and T2SP in the SD-17 project for which there is no sufficient researches and available dataset (most specifically for Bangla language) based on what we can follow approaches for training the model and get the output. Though the later one (T2SP) is a new research topic but the former one (SLR) is in race to get the expected optimal results. Similar pattern of solutions are already been implemented with acceptable amount accuracy in the broad community globally: SignAll - translating American sign automatically and DGS Corpus - produced world's biggest corpus dataset for German Sign Language.

As we have to design our application almost from the scratch, before jumping into the solution approaches we need to look around what are the probable close to match researches, algorithms and feasibility in the community for our defined problem. With that intention in mind we tried to perceive some knowledge from some researches as safety measures so that we don't waste our times reinventing the wheel that other people already did and didn't find acceptable results – meaning we can follow mostly the success areas and avoid results that below expectations.

AI team of the giant, Google, also worked upon real time sign language detection with "DGS Korpus", said as the biggest corpus globally. As we are working with the same objective of detecting sign language with addition of recognition also, perceiving a little knowledge from google AI team's research paper is a better option. "Real-Time Sign Language Detection using Human Pose Estimation" is the published title of their research paper and we gone through it. From that research paper, some references has been taken for our research exploration purpose and here below are those referred researches paper's gist:

Research Paper-1: "**Sign Language Recognition, Generation and Translation: An Interdisciplinary Perspective"** [5] , Microsoft Research - Cambridge, MA USA & Redmond, published in ResearchGate, conducted by Oscar Koller (RWTH Aachen University), Patrick Boudreault (Gallaudet University) and their team.

The research team summarizes the findings of the interdisciplinary workshop, including background information on Deaf culture in American Sign Language Community and sign language linguistics that is often overlooked by computer scientists, a review of the state-of-the-art, a list of pressing challenges, and a call to action for the research community. This paper serves to introduce computer science to readers both inside and outside of the field, highlight prospects for multidisciplinary cooperation, and assist the research community in prioritizing which challenges to tackle next, probably massive clean data!.

Our Findings: *The research team solely tried to elicit interdisciplinary overview of sign language recognition, generation, and translation. The team brought up 3 challenges.*

- *The overview of current SLR projects output.*
- *The challenges of this field.*
- *The actions or paths for computer scientist.*

These findings are actually based on American Sign Language (ASL). The vocabulary size of their datasets varies from about 100–2,000 distinct signs. ELAN & Anvil software's were used for data annotation. They experimented with two separate methods. Methods that use predefined representations are highly compatible with grammatical translation rules. On the other hand, methods that do not use such representations typically use some form of deep learning or neural networks, which learn model features (i.e. internal representations) ***that suit the problem most****.*

Research Paper-2: "**OpenPose: Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields"** [6] , published in IEEE, conducted by Z. Cao, G. Hidalgo, T. Simon, S. -E. Wei and Y. Sheikh.

In this research work, they presented a real-time multi-person 2D pose estimation is essential for machines to recognize and interpret humans and their interactions visually. At first, they offer an explicit nonparametric representation of the keypoints association, which encodes both the position and the orientation of human limbs. Second, they create an architecture that learns part detection and association at the same time. Third, they show that a greedy parsing algorithm is suitable for producing high-quality body position that parses while maintaining efficiency regardless of the number of persons. Fourth, they brought up that PAF refinement is considerably more relevant than PAF and body part location refinement, when it comes to runtime precision. Fifth, they exemplified that merging body and foot estimates into a single model improves each component's accuracy while reducing inference time when run consecutively. To do so, they have built a foot keypoint dataset with 15K foot keypoint instances that we'll make available to the public. Finally, they released OpenPose, the first real-time system for detecting keypoints on the body, foot, hand, and face.

Many academic subjects requiring human analysis, such as human re-identification, retargeting, and Human-Computer Interaction, are now widely used in the library. Additionally, ***OpenPose is now part of the OpenCV library***.

Our Observations**:** *This paper revealed about real-time human 2d pose estimation based on anatomical keypoints. It is now an open-source real-time system for multi-person 2D pose detection, including body, foot, hand, and facial keypoints. There are two notable approaches here. Frist one is the top-down approach that directly leverage existing techniques for single-person pose estimation. The second one is the bottom-up approach that do not directly use global contextual cues from other body parts. In this process, each image is analyzed by a CNN for generating a set of feature map. This CNN is initialized by the first 10 layers of VGG-19. They evaluated their method on three benchmarks for multi-person pose estimation: (1) MPII human multi-person dataset, 2) COCO keypoint challenge dataset and 3) their foot dataset. They also analyzed their failure cases and improved with some rotation augmentation.*

Research Paper-3: "**Neural Sign Language Translation**" [7] , Necati Cihan Camgoz1, Simon Hadfield, Oscar Koller, Hermann Ney, Richard Bowden from University of Surrey, UK and RWTH Aachen University, German jointly carried out this research and published in CVPR and IEEE also.

In this empirical work, the research team introduced a difficult topic of Sign Language Translation and proposed the first end-to-end solution. They utilized a machine translation approach, treating sign language as a totally independent language and recommending SLT (Sign Language Translation) rather than SLR (Sign Language Recognition) as the genuine path to enable communication with the deaf, in contrast to earlier research. To learn to align, detect, and translate sign videos to spoken text, they combined CNN-based spatial embedding, several tokenization methods, including state-of-the-art RNN-HMM hybrids, and attention-based encoder-decoder networks.

To test the method, they used the first publicly available continuous sign language translation dataset, PHOENIX14T. They carried out a number of experiments and came up with a number of recommendations for future research. It may also be able to infuse specific intermediary subunit information via a method similar to SubUNets, bridging the gap between S2T and S2G2T.

***Our Findings*:** *This paper worked on grammatical and linguistic structures of sign language which differ from spoken language. They aimed to generate spoken language translations from sign language videos, taking into account the different word orders and grammar. They provided the first publicly available continuous SLT dataset, RWTH-PHOENIX-Weather 2014T, and explored the video to text SLT problem. According to this paper, the first important factor is that collection and annotation of continuous sign language data is a laborious task. The PHOENIX datasets were created for CSLR and they provide sequence level gloss annotations. These datasets quickly became a baseline for CSLR. They followed seq2seq learning approaches in which the objective is to learn a mapping between two sequences. They propose delegating encoding and decoding to two separate RNNs.*

Research Paper-4: "**Automatic Sign Language Identification**" [8] , Author Peter Wittenburg and Tom Heskes proposed a Random-Forest based sign language identification system and published their research work in IEEE International Conference.

The suggested sign language identification system (SLID) is appealing due to its simplicity-it relies on low-level visual cues rather than phonetic transcription and good performance - it uses a random forest algorithm. The technique works with a 95 percent accuracy rate (F1-score). We may draw one essential conclusion from this performance: sign languages, like written and spoken languages, can be identified using low-level features.

***Our Observations*:** *This paper proposed Random-Forest based approach and gained 95% accuracy by evaluating British SL & Greek SL. They used low-level visual features and is based on the hypothesis that sign languages have varying distributions of phonemes (hand-shapes, locations and movements). This paper does not attempt to directly use non-manual signs for language identification. They followed the Stokoe's model. The central idea of Stokoe's model is that signs can be broken down into phonemes and that the phonemes contrast only simultaneously. An alternative to Stokoe's model is that of Move-Hold model. They tested their*

sign language modeling and identification system on a part of data that is publicly accessible from the Dicta-Sign Corpus.

Research Paper-5: "**Sign Language Recognition Based On Hand and Body Skeletal Data**" [9], Dimitrios Konstantinidis, Kosmas Dimitropoulos and Petros Daras from ITI-CERTH, Thessaloniki, Greece, published their work in 3DTV-Conference in IEEE.

This work introduces a novel SLR system that proposes the extraction and processing of hand and body skeletal data from video sequences, attempting to overcome the limitations of earlier approaches. Despite problems in obtaining reliable skeletal data due to occlusions, but this SLR system outperforms existing vision-based SLR systems on the LSA64 dataset.

Our Findings: *The followed approach relies on hand and body skeletal features extracted from RGB videos and, therefore, it acquires highly discriminative for gesture recognition skeletal data without the need for any additional equipment, such as data gloves, that may restrict signer's movements. This paper fully rely on RGB features of hand and body skeletal data. They proposed a methodology that bridges the gap between direct measurement and vision-based approaches thus taking advantage of both methods and overcoming their limitations. More specifically, they propose a SLR system that is based on the processing of video sequences in order to extract accurate body and hand skeletal data that will then be employed for robust SLR. They worked on closed environment to achieve the accuracy. The optimization of the hyper-parameters that affect the performance of their proposed SLR system is performed after experimentation on the LSA64 dataset. The final network was implemented in Keras-Tensorflow framework and trained using the Adam optimizer with batch size of 32 and learning rate equal to 0.0001.*

Conclusion – Research papers sometimes theory basis mostly, empirical examples are rarely seen when it comes about hardware resources consumptions. The research papers we brought up here to follow some approaches that closely matches with our defined problem of SLR

As we went through the techniques and algorithm followed in Paper-1 and Paper-3, we found hints how to prepare and use data, it also came into light while preparing data that what parameters we need to focus on more. The papers also pointed us on how linguistic model structure should look like if the word order differs, how we need to deal with it and how to reduce the gap between static and stream data. For the stream data we will follow sequence level gloss annotation thus we can compare dataset with their baseline dataset. Also, we found one direction to follow the **seq2seq** learning approach for mapping two sequences.

Soon after getting the dataset preparation clue and sequence mapping for video stream data, Paper-2 and Paper-4 directed us to identify human posture and its skeletal geometry to consider. In Paper-2, **OpenPose** gave us information on how to detect human movement with specific keypoint analysis of the CNN based feature mapping. On the other hand, Paper-4 followed Stokoe's model and Move-Hold model which helps us to identify the distributions of phonemes from hand-shape, location, and movements.

And finally Paper-5 gives a clear vision of feature-extraction of the human body from RGB videos and how to manage highly discriminative gesture recognition. It also emphasizes on how the environment affects our model accuracy.

3.2 How Artificial Intelligence Helping Us

3.2.1 Abstract

Technologies brought by trendy Artificial Intelligence will surely play an important role in breaking down the communication barriers of deaf or hearing-impaired people within communities. Recent advancements in both sensing technologies and AI algorithms more specifically Deep learning algorithms have paved the way for the development of various applications aiming at fulfilling the needs of deaf and hearing-impaired communities. Though hard challenges are there, AI has brought state-of-the-art methods in sign language capturing, recognition, translation and representation as we saw in the research papers above. Computer Vision (CV), Machine Learning (ML) and finally Deep Learning (DL) have brought up unlimited scope in the respective fields to explore and head burring oil to reach to accuracy.

3.2.2 Computer vision

Computer Vision (CV) is the subcategory of artificial intelligence (AI) that focuses on building and using digital systems to process, analyze and interpret visual data. The goal of computer vision is to enable computing devices to correctly identify an object or person in a digital image and take appropriate action. Technically, machines attempt to retrieve visual information, handle it, and interpret results through special software algorithms. Computer vision uses convolutional neural networks (CNNs) to processes visual data at the pixel level and deep learning recurrent neural network (RNNs) to understand how one pixel relates to another.

Computer vision has enabled object detection and object tracking including skeletal and posturing geometry that has extended this field to near to close prediction. Deep learning algorithms for computer vision deals with 2D as well as 3D format. 3D vision allows building 3D point cloud – a representation of an image in the 3D format. This way, computers can catch the location and shape of an object – this directed us to a new door to open on hand geometry reading, gesture and posture reading that signals we can read silent language which is sign language!

3.2.3 Machine Learning

Machine Learning (ML) is everywhere. This is how Netflix knows which show we want to watch next or how Facebook recognizes our friend's face in a digital photo. Or how a customer service representative will know if you'll be satisfied with their support before you even take a CSAT survey. The thing about traditional **Machine Learning** algorithms is that as complex as they may seem, they're still machine like. They need lot of domain expertise, human intervention only capable of what they're designed for. In addition to that, whence it comes about the image recognition or image recognition from motion videos – ML holds some sort of flaws in accuracy. For AI designers and the rest of the world, that's where deep learning holds a bit more promise.

3.2.4 Deep Learning

Deep Learning (DL) is a subset of Machine Learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. More elaborately, a deep learning technique learn categories incrementally through its hidden layer architecture, defining low-level categories like letters first then little higher-level categories like words and then higher-level categories like sentences. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers. For our proposed solution we will use AI in terms of Machine Learning and Deep Learning.

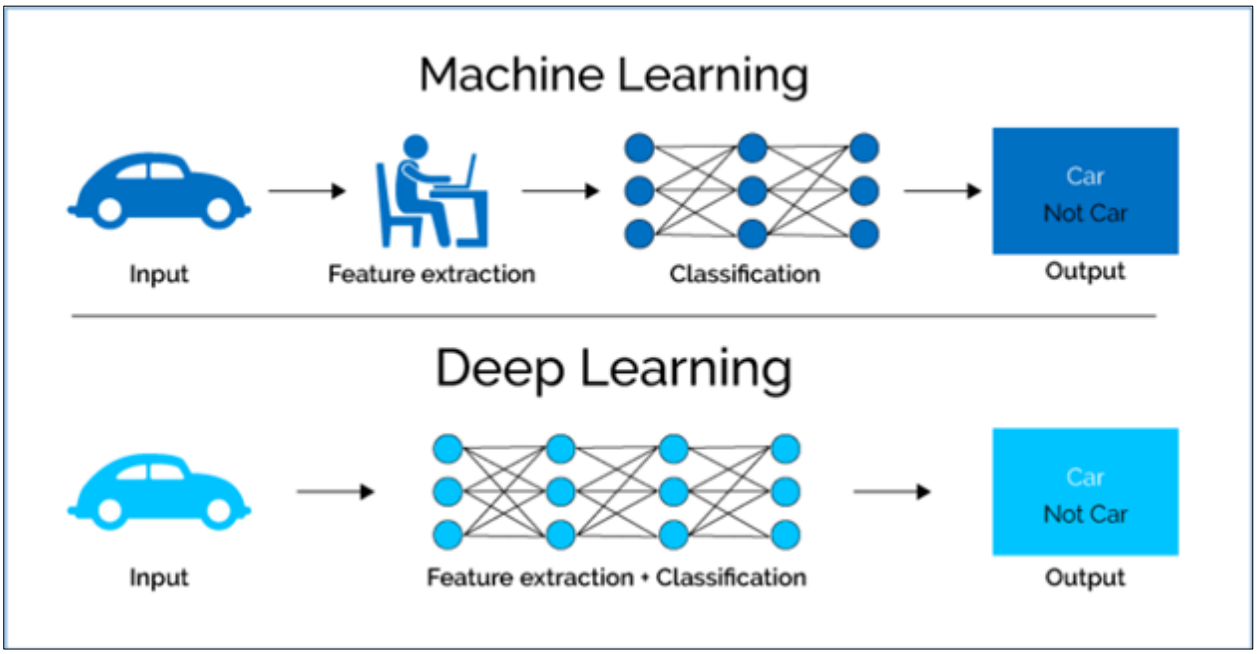


Figure 1. Machine Learning and Deep Learning Comparison

A relation between Artificial Intelligence, Machine Learning, Deep Learning, and Computer Vision can be seen closely with below diagram

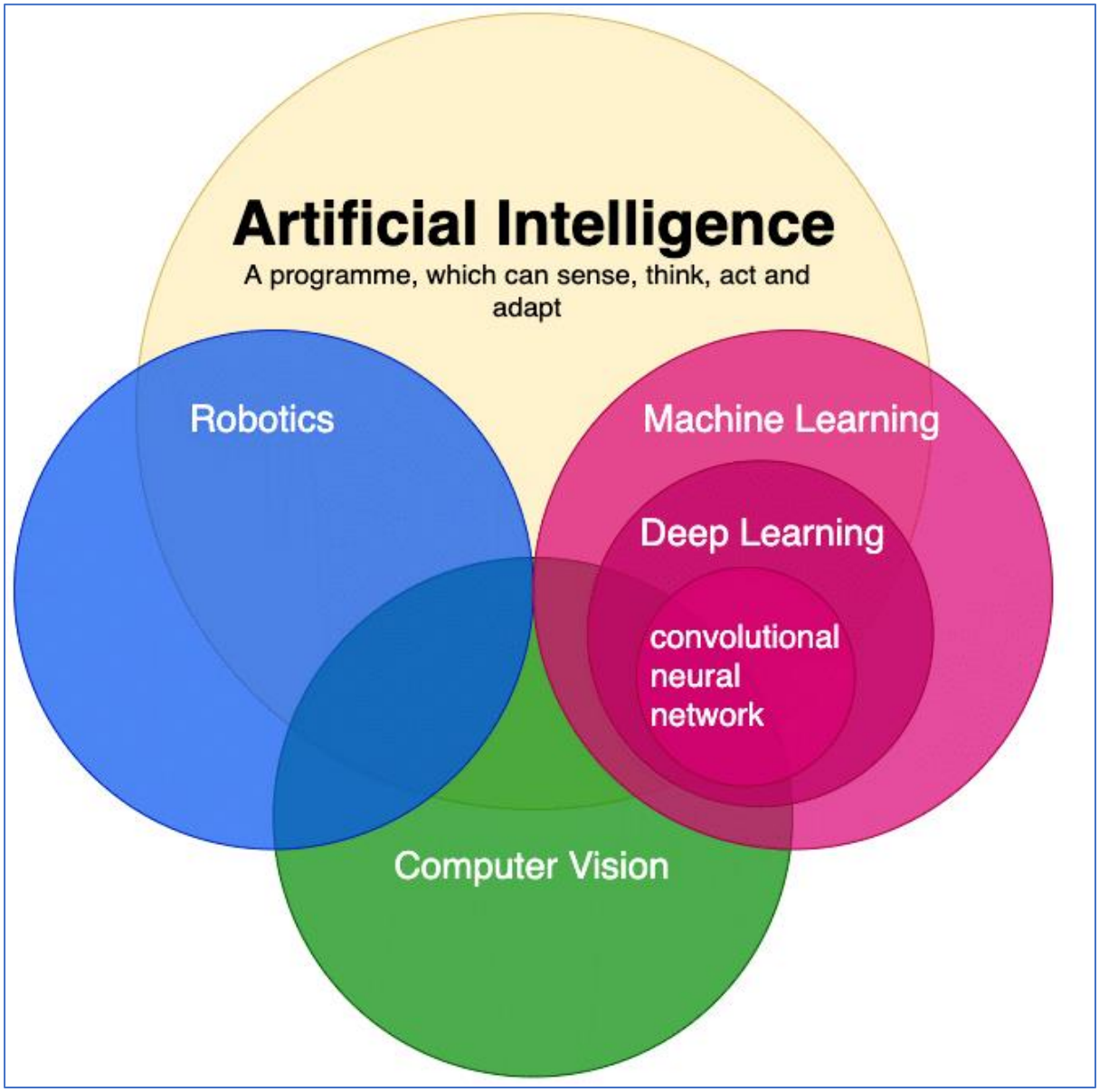


Figure 2. Relation between Artificial Intelligence, Machine Learning, Deep Learning, and Computer Vision

3.2.5 Deep Learning Framework

As of the proposed project scope, for development and implementation of BDSLR – we will heavily rely one Deep Learning, and hence we need to choose better classification approaches for optimal output with standard accuracy and hence we need to choose solid framework those are already proven in the community. Some popular deep learning framework/library will be used for the targeted problems are: **TensorFlow**, **Keras**, **Theano**, DL4j, **Caffe**, Chainer, Microsoft CNTK.

3.3 Approach for Bangla Sign Language Recognition (SLR)

As we have understood some researches (Microsoft Academic Research, Google Scholars etc.) studied some journals and conference papers to identify the best approach for a Sign Language recognition and translation system, the best steps will be followed to achieve our goal.

3.3.1 Data Preparation from Natural Bangla Language

The flow diagram shows the data, sign gloss preparation steps from natural languages:

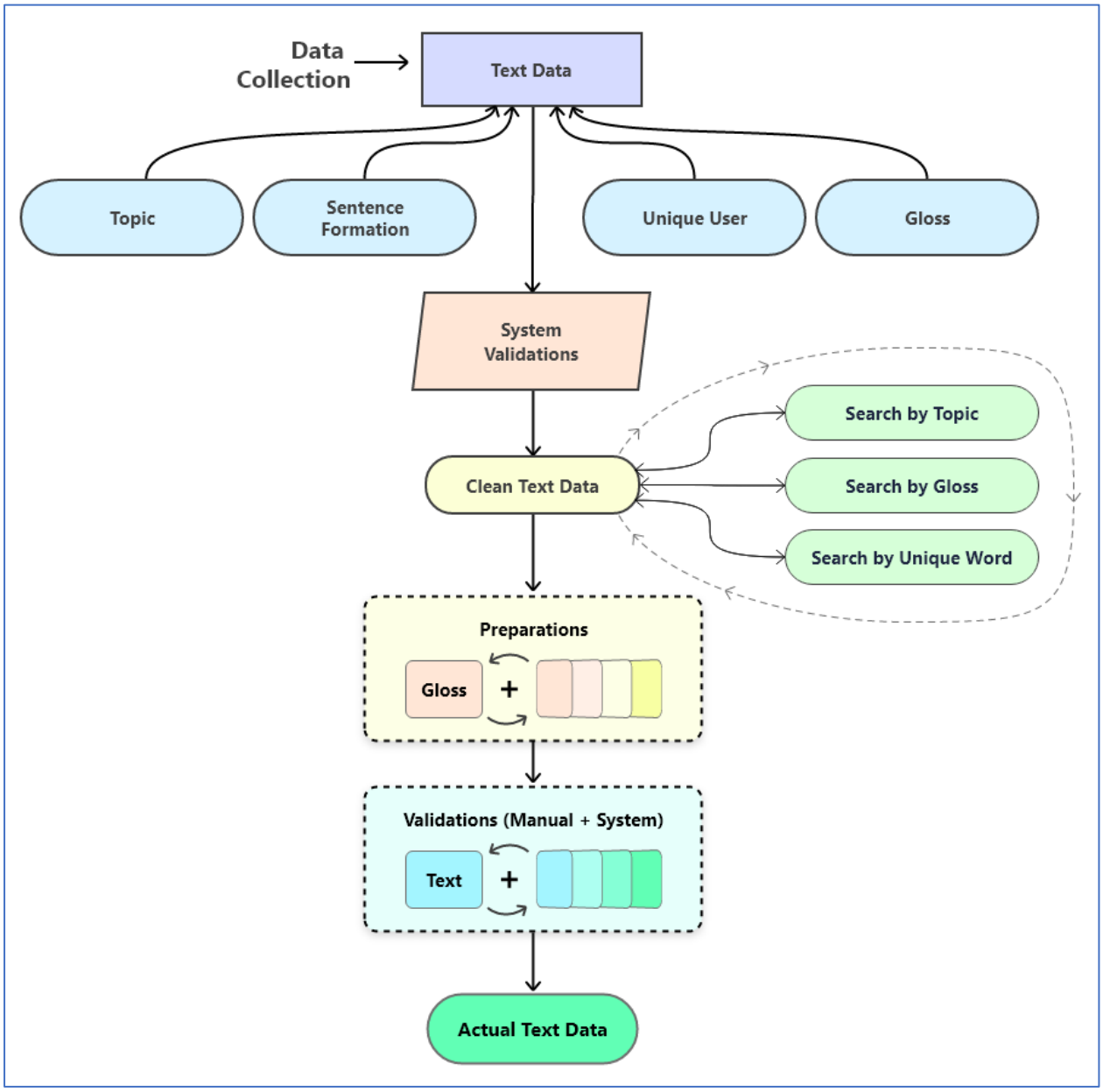


Figure 3. Preparation of Text Data with Gloss

Script Preparation: Script preparation task comprise preparing sentences those will be used for recording videos of sign gloss. Considering around 70 Bangla topics based on skill level (beginner, Medium & Advanced), 2 actors (male & female), almost 358 sentences per topic and variation of environment per actor, combination of camera, positions & act repetition. Sentences type must consider within the following categories:

Sentence	
Type 1	**Type 2**
Simple	Assertive Sentence
Compound	Interrogative Sentence
Complex	Imperative Sentence
N/A	Optative Sentence
N/A	Exclamatory Sentence

Table 1. Grammar Based Sentence Category

Topics			
SL.	**Beginner**	**Medium**	**Advanced**
1	Greetings	Introducing others	Emergency situations
2	Introducing self	Encouraging words	Descriptions
3	Introducing people	Buying and selling	Comparing things
4	Identifying people, things	National numbers and prices	Questions and expressions with time
5	Classroom questions	International numbers and prices	Agreeing/disagreeing
6	Asking for information	Asking for favors	Asking permission
7	Giving information	Asking for repetition	Discussing Sensitive Topics
8	Simple sentences	Requesting	Accepting and Refusing
9	Simple questions	Inviting	Supporting opinions
10	Numbers and counting	Offering	Classifying
11	Talking about family	Talking about abilities	Clarifying
12	Talking about favorite things	Expressing possibility	Simple presentations
13	Talking about here and now	Talking about locations	Job interviews

Table 2. Skill based Sentence, Part 1

Topics			
SL.	**Beginner**	**Medium**	**Advanced**
14	Describing people	Asking for directions	
15	Telling time	Giving directions	
16	Talking about past actions	Asking about place/location	
17	Talking about the future	Talking about travel	
18	Talking about life events	Count and non-count nouns in context	
19	Talking about feelings/health	Narrating	
20	Expressing likes and dislikes	Talking at police station	
21	Simple shopping	Talking with doctor	
22	Short questions and answers		
23	Closing a conversation		
24	Expressing thanks		
25	Situation: At the store		
26	Talking about occupations		

Table 3. Skill based Sentence, Part 2

SL.	Signs
1	Alphabets & Number
2	Family & Relatives
3	Relationships
4	Food & Drinks
5	Dresses/Attires
6	Household Objects
7	Disease & Treatment
8	Education
9	Religion
10	Sports
11	Transportation
12	Social Custom & Occasion
13	Business
14	Profession
15	Cosmetics
16	Human Characteristics
17	Nature and Environment
18	Geographical Elements
19	Animals
20	Science and Technology
21	Information and Communication
22	Time
23	International Affairs
24	Anatomy
25	Crime and Law
26	Institutional Words
27	Grammar

Table 4. Special Sign based Word/Sentence

With all of the possible combinations, a target has been taken to produce around 20K to 25K unique sentences, a parallel task though, along with preparing sentences, running sign gloss will be prepared also based on what video capturing will be performed. For better identification and manageability sentences will be marked with a unique number and the format will be as follows:

<Project Name><DataSet><Skill Level><Topic Number, starts with "T"> <Sentence Number starts with "S">
Example: <SD17><DS><B><T19><S1> or "**SD17DSBT19S1**"

3.3.2 Running Gloss Validations from Two Actors

As we have natural language script ready with running sign gloss, actors, who will perform the sign language for capturing video, will follow the combination as described below:

- Topic "Health" will combine with a sign "Number" and form a "Simple" type "Assertive" sentence.
- Same topic "Health" will combine with a sign "Number" and form a "Complex" type "Interrogative" sentence.
- Again, topic "Health" will combine with a sign "Food & Drinks" and form a "Simple" type "Interrogative" sentence.

Along with above mentioned combinational pattern, aligning with the ready sentences, two types of running sign gloss will be collected from two sign actors. This will bring the variation in sign gloss and also will help to validate if one actor giving the right sign gloss or not. An example below shows that two actors have given running sign gloss for single natural sentence:

Sl.	Sign	ID	Standard Bangla	Running Sign Gloss - Miss Lata	Running Sign Gloss - Mr. Arif
1	Number	SD17DSAT19S1	তোমার ওজন কত?	তোমার ওজন কত? / তোমার শরীরের ওজন কত?	তোমার শরীর ওজন কি?
2	Number	SD17DSAT19S2	আমার ওজন ৬০ কেজি	আমার ওজন ৬০ কেজি	আমার শরীর ওজন ৬০ কেজি
3	Food & Drinks	SD17DSAT19S3	সুস্বাস্থ্যের জন্যে নিয়মিত সবজি খাওয়া উচিত	নিয়মিত সবজি খাওয়া স্বাস্থ্য ভালো হবে	স্বাস্থ্য ভালো দরকার প্রতিদিন সবজি খাও
4	Food & Drinks	SD17DSAT19S4	কিছু স্বাস্থ্যকর খাবার কি কি?	স্বাস্থ্য ভালো খাওয়া নাম কি কি?	স্বাস্থ্য ভালো দরকার খাওয়া কি?
5	Disease	SD17DSAT19S5	আমার মাথা ব্যাথা করছে	আমার মাথা ব্যাথা	আমার মাথা ব্যাথা আছে
6	Disease	SD17DSAT19S6	বিকাল থেকে তার মাথা ঘুরছে	বিকাল থেকে তার মাথা ঘোরা	বিকাল মাথা ঘুরা এখন মাথা ঘুরা আছে
7	Disease	SD17DSAT19S7	হাত কেটে গেছে	হাত কাটা	হাত কাটা

Following the procedures above, targeted 20k-25K sentences and running sign gloss will be prepared and validated. This script will be marked with unique ID as discussed above. Finally, from the ready script of natural sentences and their equal running gloss we can extract the following four parameters:

- Topic
- Sentence formation
- Unique word
- Gloss

The table below shows topic wise standard Bangla sentences with their corresponding running sign gloss that will be used for preparing video sign data to feed to machine:

Sl.	Sign	ID	Standard Bangla	Running Sign Gloss - Lata	Running Sign Gloss - Arif	Notes	Gloss per sentence	Type
1	Education	SD17DSCT13S1	আপনার শিক্ষা সম্পর্কে আমাদের বলুন	তুমি পড়া কি বলো	আপনার পড়া কি? আমাদেরকে বলো?		4	Simple
2	Transportation	SD17DSCT13S2	অফিস থেকে পরিবহন সুবিধা পাবেন, কিন্তু ৩ মাস পর	অফিস যাওয়া আসা গাড়ি দিবে, কিন্তু ৩ মাস পর	অফিস গাড়ি সুবিধা দিবে, কিন্তু ৩ মাস পর		9	Compound
7	Family & Relatives	SD17DSCT13S7	আপনার পরিবারে কে কে আছেন?	তোমার পরিবার কে কে আছে	আপনার পরিবার মা বাবা ভাই বোন কি কি বলো?		5	Exclamatory Sentence
8	Family & Relatives	SD17DSCT13S8	আপনি কি বিবাহিত?	তুমি বিয়ে হয়েছে?	আপনি বিয়ে? / আপনি বিয়ে শেষ?		3	Exclamatory Sentence
9	Number	SD17DSCT13S9	জী, আমি বিয়ে করেছি ২ বছর হয়েছে	হ্যাঁ, আমি বিয়ে হয়েছে দুই বছর	হ্যাঁ ২ বছর আগে আমি বিয়ে শেষ		7	Assertive Sentence
10	Number	SD17DSCT13S10	এই পেশায় আমার কাজের অভিজ্ঞতা ৫ বছর	এই কাজ আমি ৫ বচর পুরাতন / এই কাজ আমি ৫ বচর অভিজ্ঞ আছে	এ কাজ আমি ৫ বছর জ্ঞান/অভিজ্ঞতা আছে		7	Simple
11		SD17DSCT13S11	আপনার পরিচয় দিন	তোমার নাম বলো	আপনার নাম কি?			
12		SD17DSCT13S12	আমার নাম আরিফুল	আমার নাম A R I F U L	আমার নাম A R I F U L	আরিফুল = ARIFUL, ফিঙ্গার স্পেলিং। They use **English** alphabet for convenience		
13		SD17DSCT13S13	আপনার সম্পর্কে বলুন	তোমার নাম, পড়া, পরিবার, নানা বিষয় বলো / তোমার জীবন বলো	আপনার নিজ বলো			

Figure 4. Sentences example with formation of gloss from topic "Health"

3.3.3 Preparing Video Data from Text Script

A studio has been setup with highly equipped multi-sync cameras, Video Artist/Shooting Staff, Video editors and required human resources for the same has been already deployed. From the ready script, based on the video specifications (all the parameters defined by the client) different combinations recording has been started for capturing video stream for each sentence.

Figure 5. Studio Setup and Mock-up sign performing is in progress

Video shooting staff compositing - the team will work with one director of photography, two assistant directors, two editors, four camera men and two light man and other associates. The table below shows the team composition with their roles:

Name of Staff	Position Assigned
Chandan Roy Chowdhury	Director of Photography (DOP)
Moniruzzaman Monir	Assistant Director # 01
Soroar Rana	Assistant Director # 02
Mark Biplob	Editor # 01
Mac Khokon	Editor # 02
Laku Komal	Cameraman # 01
Shawon Mridha	Cameraman # 02
Bishojeet Borua	Cameraman # 03
Raju Raj	Cameraman # 04
Rajib Rahman	Light man # 01
Dawed Karim	Light man # 02

Table 5. Video Shooting Staff

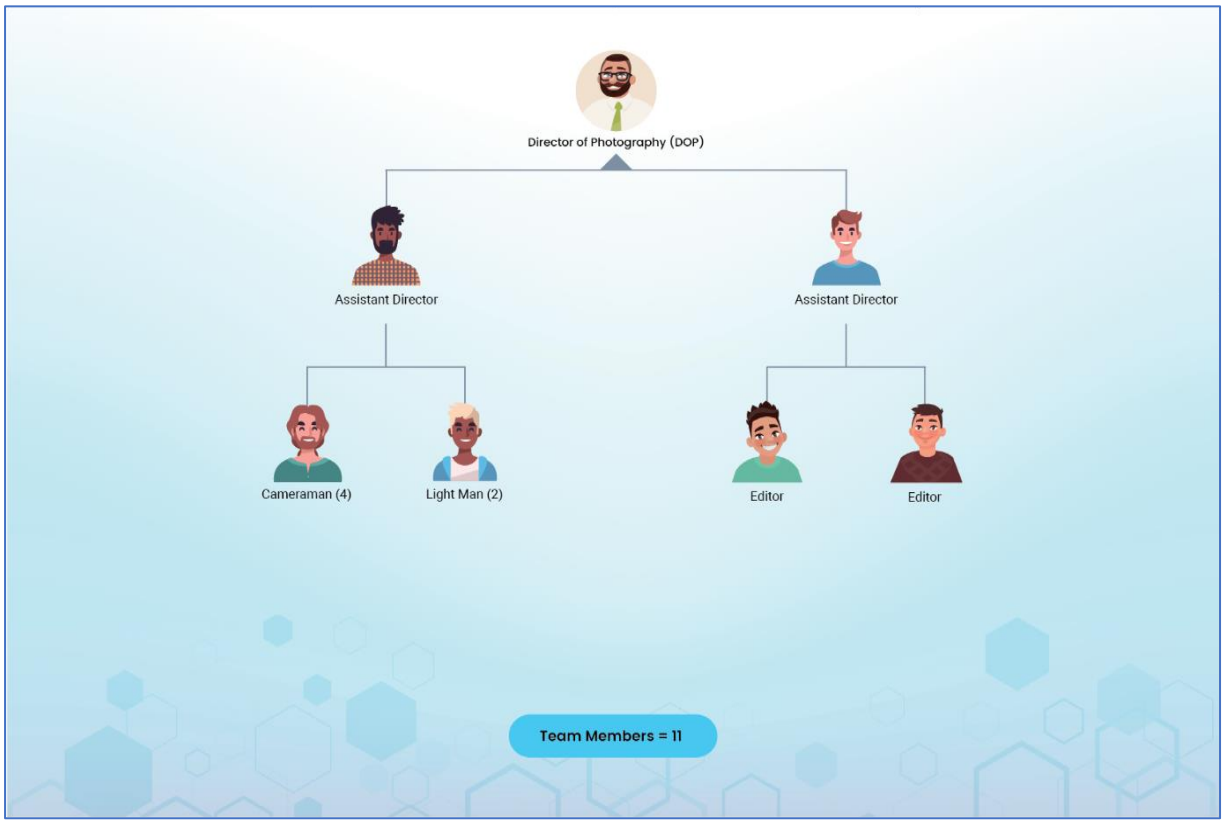


Figure 6. The organogram of Studio Staff

Recording Arrangement

Camera Details: Sony a7R III Mirrorless Camera: 42.4 MP Full Frame High Resolution Interchangeable Lens Digital Camera with front end LSI Image processor, 4K HDR Video and 3" LCD screen - ILCE7RM3/B Body.

Camera Position: The video data must be taken from camera 4 perspective. Those are –

- From front
- From above
- From angle 45^{o}
- From angle 90^{o}

Actors: There will be 3 different age-shaped actors containing male and female.

Resolutions: The video which has been prepared from text script has the RGB resolution of 1920*1080 and depth resolution of 512*424.

Background: There was 3 type of background light source and color temperature for the prepared video. Original, bright and dim is 3 types of light source.

Sentence/word recording: As we have defined all of the aspects of recording, its assumable that there will be 54 environment per actor, combination of camera, positions & act repetition. Here is an example of 54 combination of environment for one actor –

	Environment	Sentence	Camera	Position	Repetition	Done	Case
4	E1 = original light	SD17T1S1	C1 = front cam	P1 = original position	1		
5	E1 = original light	SD17T1S1	C1 = front cam	P1 = original position	2		
6	E1 = original light	SD17T1S1	C1 = front cam	P1 = original position	3		
7	E1 = original light	SD17T1S1	C1 = front cam	P2 = far position	1		
8	E1 = original light	SD17T1S1	C1 = front cam	P2 = far position	2		
9	E1 = original light	SD17T1S1	C1 = front cam	P2 = far position	3		
10	E1 = original light	SD17T1S1	C1 = front cam	p3 = near position	1		
11	E1 = original light	SD17T1S1	C1 = front cam	p3 = near position	2		
12	E1 = original light	SD17T1S1	C1 = front cam	p3 = near position	3		
13	E1 = original light	SD17T1S1	C2 = 45 deg	P	1		
14	E1 = original light	SD17T1S1	C2 = 45 deg	P	2		
15	E1 = original light	SD17T1S1	C2 = 45 deg	P	3		
16	E1 = original light	SD17T1S1	C3 = 90 deg	P	1		
17	E1 = original light	SD17T1S1	C3 = 90 deg	P	2		
18	E1 = original light	SD17T1S1	C3 = 90 deg	P	3		
19	E1 = original light	SD17T1S1	C4 = above cam	P	1		
20	E1 = original light	SD17T1S1	C4 = above cam	P	2		
21	E1 = original light	SD17T1S1	C4 = above cam	P	3		
22	E2= dim light	SD17T1S1	C1 = front cam	P1 = original position	1		
23	E2= dim light	SD17T1S1	C1 = front cam	P1 = original position	2		
24	E2= dim light	SD17T1S1	C1 = front cam	P1 = original position	3		
25	E2= dim light	SD17T1S1	C1 = front cam	P2 = far position	1		
26	E2= dim light	SD17T1S1	C1 = front cam	P2 = far position	2		
27	E2= dim light	SD17T1S1	C1 = front cam	P2 = far position	3		
28	E2= dim light	SD17T1S1	C1 = front cam	p3 = near position	1		
29	E2= dim light	SD17T1S1	C1 = front cam	p3 = near position	2		
30	E2= dim light	SD17T1S1	C1 = front cam	p3 = near position	3		
31	E2= dim light	SD17T1S1	C2 = 45 deg	P	1		Case SD17T1C1
32	E2= dim light	SD17T1S1	C2 = 45 deg	P	2		
33	E2= dim light	SD17T1S1	C2 = 45 deg	P	3		
34	E2= dim light	SD17T1S1	C3 = 90 deg	P	1		
35	E2= dim light	SD17T1S1	C3 = 90 deg	P	2		
36	E2= dim light	SD17T1S1	C3 = 90 deg	P	3		
37	E2= dim light	SD17T1S1	C4 = above cam	P	1		
38	E2= dim light	SD17T1S1	C4 = above cam	P	2		
39	E2= dim light	SD17T1S1	C4 = above cam	P	3		
40	E3= bright light	SD17T1S1	C1 = front cam	P1 = original position	1		
41	E3= bright light	SD17T1S1	C1 = front cam	P1 = original position	2		
42	E3= bright light	SD17T1S1	C1 = front cam	P1 = original position	3		
43	E3= bright light	SD17T1S1	C1 = front cam	P2 = far position	1		
44	E3= bright light	SD17T1S1	C1 = front cam	P2 = far position	2		
45	E3= bright light	SD17T1S1	C1 = front cam	P2 = far position	3		
46	E3= bright light	SD17T1S1	C1 = front cam	p3 = near position	1		
47	E3= bright light	SD17T1S1	C1 = front cam	p3 = near position	2		
48	E3= bright light	SD17T1S1	C1 = front cam	p3 = near position	3		
49	E3= bright light	SD17T1S1	C2 = 45 deg	P	1		
50	E3= bright light	SD17T1S1	C2 = 45 deg	P	2		
51	E3= bright light	SD17T1S1	C2 = 45 deg	P	3		
52	E3= bright light	SD17T1S1	C3 = 90 deg	P	1		
53	E3= bright light	SD17T1S1	C3 = 90 deg	P	2		
54	E3= bright light	SD17T1S1	C3 = 90 deg	P	3		
55	E3= bright light	SD17T1S1	C4 = above cam	P	1		
56	E3= bright light	SD17T1S1	C4 = above cam	P	2		
57	E3= bright light	SD17T1S1	C4 = above cam	P	3		
58	Combination	54					

Figure 7. 54 combination of environment for one actor

With this combinations of different environments, for each sentence, 3 actors will continue to perform the sign language video recording until our targeted sentences are completed.

Two operations will be performed on the recorded video data to identify or tokenize sentence (if multiple sentences are there in a single video) and extracting video sign gloss. These are described below:

- **Sentence Tokenizer** – This is a full sentence where multiple words will makes by grammar form and express a meaning. From this tokenizer, sentence n-gram forms which is a running sign gloss.
- **Word n-gram or unigram**- Word n-gram will help to filter the unique word from the running sign gloss.

The unique word that we will receive from unigram process is our specific gloss. Now we will identify videos with 2 contents. One content will be the sentence tokenizer and the other one will be unique word.

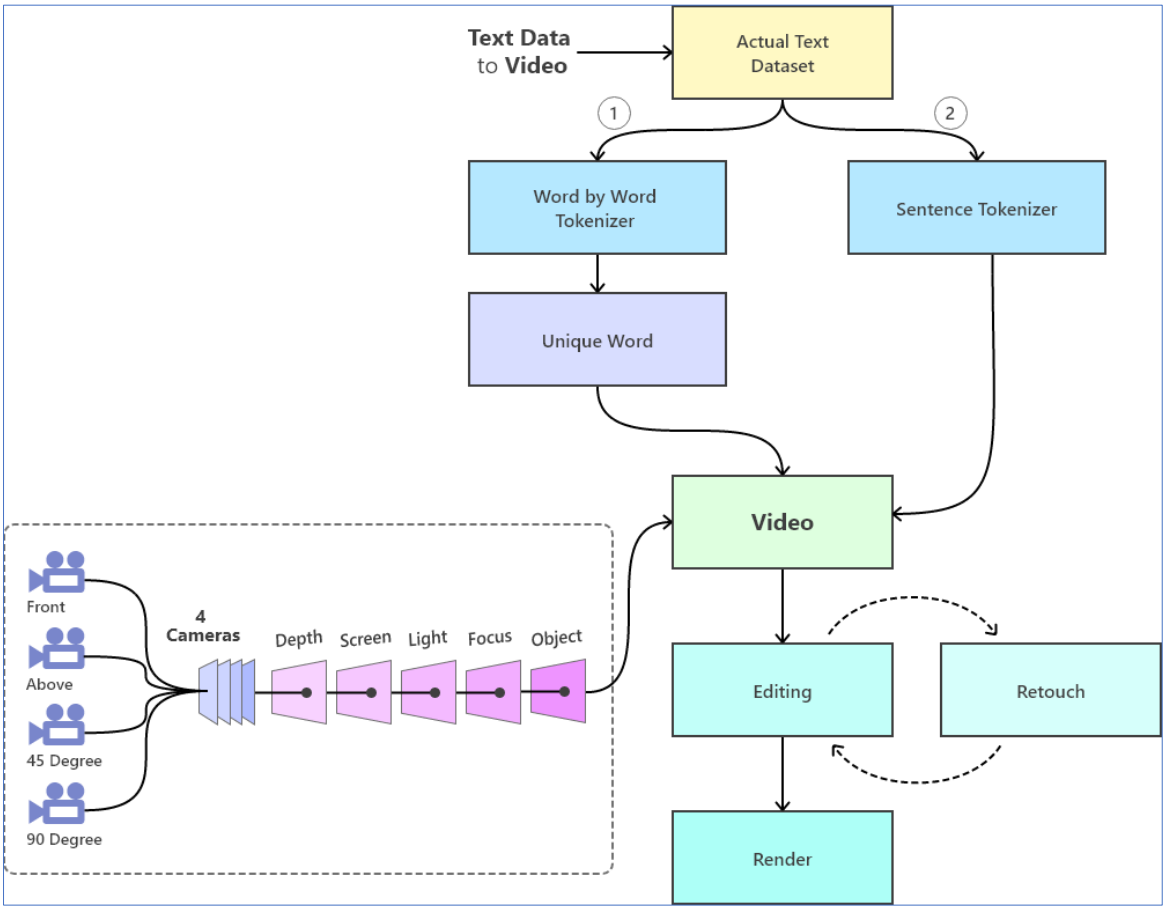


Figure 8. Text Data to Video process

The reason behind shooting videos from 4 perspective is to ensure the green background, depth sensor, light ratio, and object focus. Once the video has been taken out, it will be edited by re-touch or some mode. Lastly, we get our perfect video for rendering and rendered video for annotation tools.

3.3.4 Annotating Recorded Video Data

ELAN - EUDICO Linguistic Annotator: ELAN is computer software, a professional tool to manually and semi-automatically annotate and transcribe audio or video recordings. It has a tier-based data model that supports multi-level, multi-participant annotation of time-based media. It is applied in humanities and social sciences research (language documentation, sign language and gesture research) for the purpose of documentation and of qualitative and quantitative analysis. It is distributed as free and open-source software under the GNU General Public License, version 3.

ELAN is a well-established professional-grade software and is widely used in academia. It has been well received in several academic disciplines, for example, in psychology, medicine, psychiatry, education, and behavioral studies, on topics such as human computer interaction, sign language and conversation analysis, group interactions, music therapy, bilingualism and child language acquisition, analysis of non-verbal communication and gesture analysis, and animal behavior.

With ELAN a user can add an unlimited number of textual annotations to audio and/or video recordings. An annotation can be a sentence, word or gloss, a comment, translation or a description of any feature observed in the media. Annotations can be created on multiple layers, called tiers. Tiers can be hierarchically interconnected. An annotation can either be time-aligned to the media or it can refer to other existing annotations. The content of annotations consists of Unicode text and annotation documents are stored in an XML format (EAF). ***The beauty of the ELAN is we can do customization when needed to get the best output***.

Main features and characteristics of ELAN:

- Provides several ways to view the annotations, each view is connected to and synchronized with the media timeline
- Supports creation of multiple tiers and tier hierarchies
- Supports Controlled Vocabularies
- Allows linking of up to 4 video files with an annotation document
- Media support
    - Builds on existing, native media frameworks, like Windows Media Player, QuickTime or VLC
    - Support for audio and video formats depends on operating system, high performance media playback can usually be achieved
- Technical
    - Written in the Java programming language
    - Distributions available for Windows, macOS and Linux
    - Open source, the sources are available under a GPL 3 license

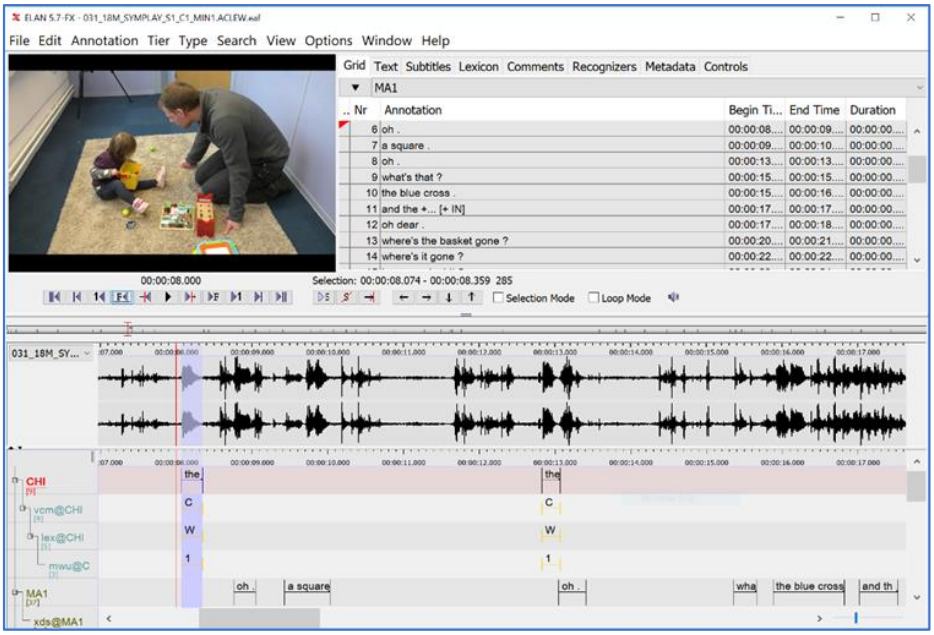


Figure 9. Screenshot from ELAN

Ilex: iLex is a tool for sign language lexicography and corpus analysis that combines features found in empirical sign language lexicography and in sign language discourse transcription. It supports the user in integrated lexicon building while working on the transcription of a corpus and offers a number of unique features considered essential due to the specific nature of signed languages. iLex binaries are available for macOS. It generates iLax formatted data as annotation.

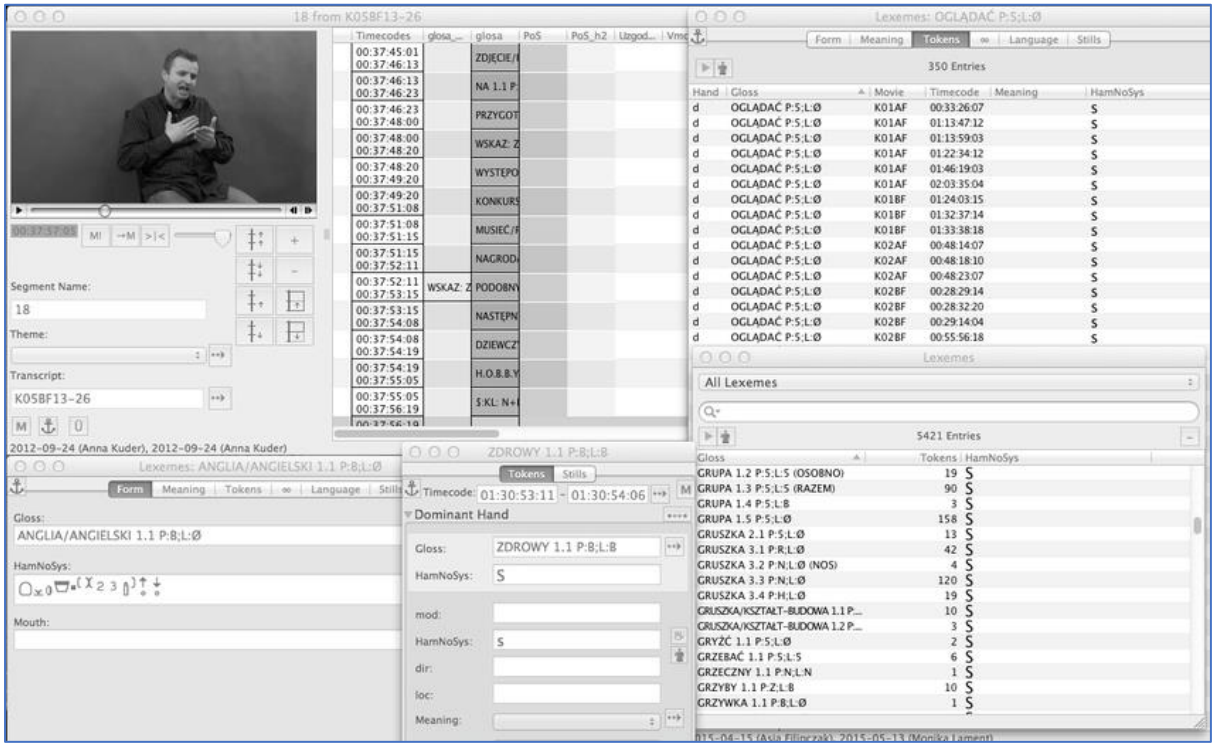


Figure 10. Annotating

SignStream: SignStream is a tool for linguistic annotations and computer vision research on visual-gestural language data SignStream installation is only available for old versions of MacOS and is distributed under MIT license. It generates XML as output.

System requirements:

- Mac OS 10.8 through 10.14. Important warning: SignStream® is not compatible with the latest version of the Mac operating system. We hope to resolve compatibility issues in the near future, but as of now, SignStream® will not work with Mac OS versions 10.15 or later.
- Java 6 (legacy) is also required

Video codecs that work with the application; if your video is in a different format, you should first convert it to one of these:

- MPEG-4 (.mp4)
- AVC1 (.mp4)
- MPEG-4 (.mov)
- H.264 (.mov)

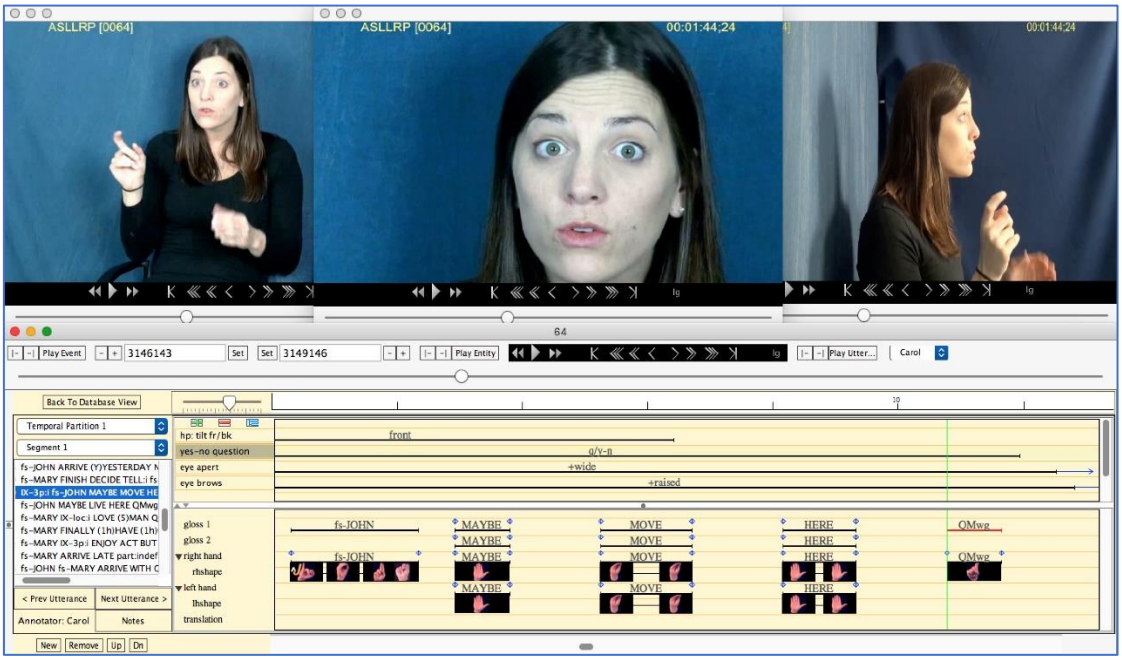


Figure 11. SignStream

Prepared video data will be fed to ELAN to complete the annotations. In this process, we will use annotated data for better performance to train the model. If performance is not satisfactory, we can go back for re-annotate.

The above process of annotation is driven by several steps. Firstly, if we consider 4 rendered video for each sentence collected from previous phase which was “Preparation of video data

from text script" and unique words as gloss. Now we will upload the unique words along with those 4 videos which will be synced in our customized annotation tools.

Secondly, a Pose model (OpenPose) will be implemented by manually in annotation tool with instruction of 5sec video mining which will generate posture key points. On other side, we will select gloss from the 5sec video which depends upon **mouth** and **hand** gestures. All these selected gloss will generate coordinate and timeline. Both of the process, generating key points and gloss selection, will run simultaneously.

Finally, we will save those posture key points, generated coordinate and timeline which is the output of annotation tools as annotated dataset. This output will be used to train the model. The flow of the Annotation of recorded video is as follows:

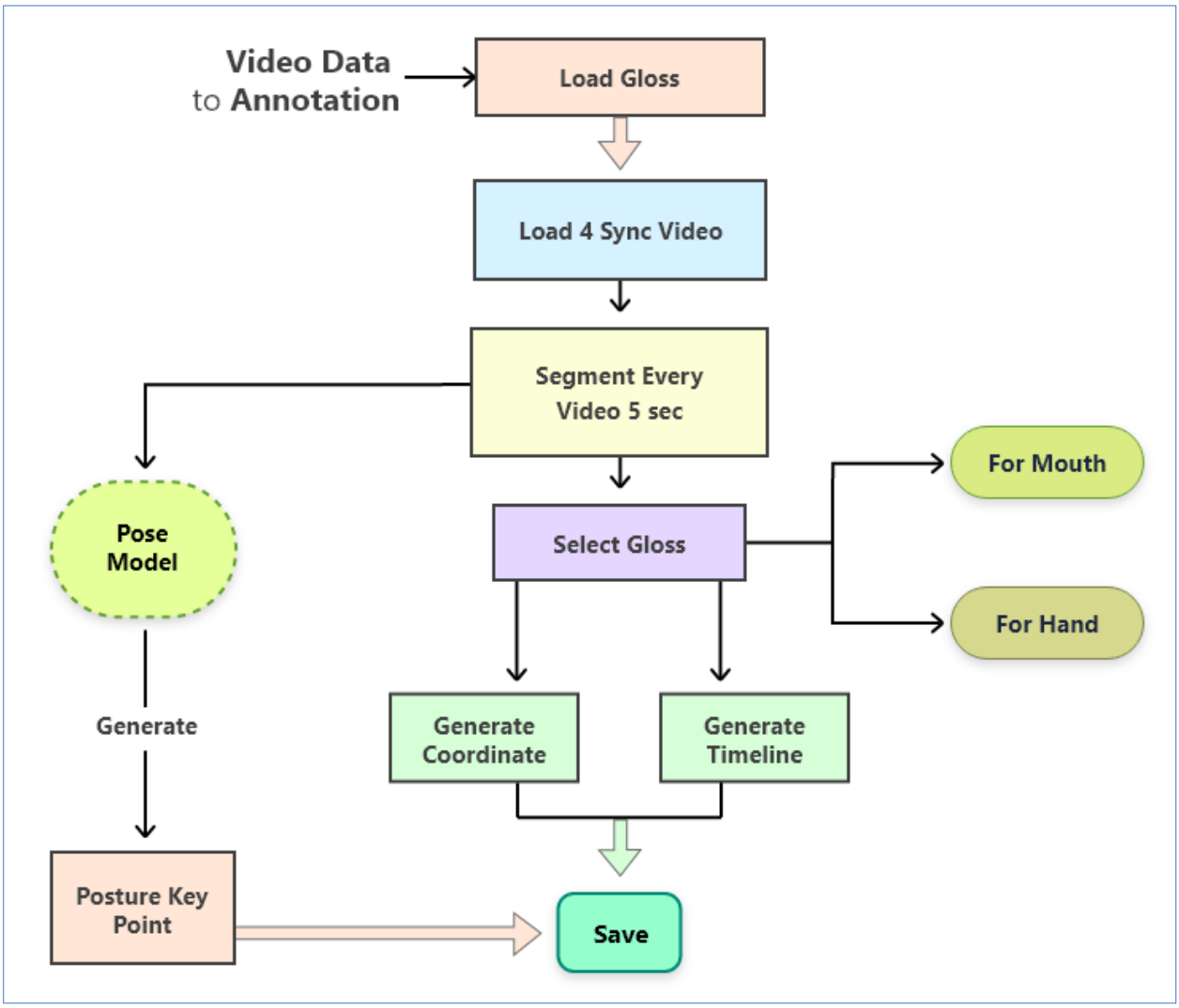


Figure 12. Video Data to Annotation

BdSL Corpus

BdSL corpus will be a scientifically edited collection of examples of how Bangla sign language is commonly used. BdSL corpus, for example, will contain a large number of written texts, or

recorded or filmed conversations. Such data collections are used to explore the usage of a sign language or to find out about the vocabulary and grammar of this language.

One of the aims of the BdSLR project is to develop a corpus that represents the everyday speech of competent Bangladeshi Sign Language (BdSL) users. For this purpose, a vast data collection of different category of people (age, regions, dialects etc.) will be carried out. The data collection will consist several parts:

1. Monologue (single person conversation)
2. Dialogue (Two people conversation)
3. Group discussions/Debates
4. Discussing daily life
5. Discussing professional career
6. Discussing movie and movie thereafter etc.

This amount of data corresponds to approximately 3 million usages (occurrences) of signs, which is comparable to corpora of spoken languages. The annotation of the corpus is very time-consuming as most of the work cannot be done automatically but will tedious manual job. The BdSL corpus will contain a variety of narrations that are of interest for the deaf community from a cultural point of view. These recordings will potentially be useful as educational materials for institutions so that deaf community get accesses. In order to make this valuable material also accessible to everyone interested in BdSL and sign language, primarily narrations and conversations will be provided for publication in the BdSL Corpus.

The minimum unit for running text is sentence. So that we have to develop a sentence aligned video footage corpus. In this set, every sentence/ topic had 3 repetitions, 3 different camera views, 5 different age-shaped producers/ actors, 2 positioning/ camera-object distance, 3 environments (background-light-setup). Therefore, every sentence has 270 times combinations and every topic have 8100 (270X30) to 40500 seconds recordings. That means every topic should have video footage from 2.25 hours to 11.25 hours.

Isolated sign recognition is included in the scope as a supportive dataset for specific use of BdSL. The minimum unit for set B is sign gloss. Set B should cover regional and dialectical variation over Bangladesh.

Set A: Predefined common monologue/ dialogue/ topic (10-20 seconds/topic)

Running Sign Gloss Per sentence	Running Sign Gloss In a topic	Video Per Sentence	Video Per topic total
1-6 each	100+	4-8 Seconds	30-150 seconds

Table 6. Predefined common monologue/ dialogue/ topic

Set B: BdSL sign gloss (that cover most frequent and common glosses)

Isolated word/gloss	Total set	Video per gloss	Total Video
1 each	4000+	4-5 seconds	16000 seconds

Table 7. BdSL sign gloss

Corpus Data Structure

Transcript	Age Group	Skill Level	Topics	Video A	Video B	Video Total	SRT	Video AB	Open Pose
bdslkorpus_ber_01	18-30	Beginner	Greetings, Introducing self, Introducing people, Identifying people, things.						
bdslkorpus_ber_02	46-60	Beginner	Asking for information, Giving information, Simple sentences, and Simple questions.						
bdslkorpus_ber_03	46+	Beginner	Numbers and counting, Talking about family, Talking about favorite things, Talking about here and now.						
bdslkorpus_ber_04	31-60	Medium	Encouraging words, Buying and selling, National numbers and prices						
bdslkorpus_ber_05	61+	Medium	Asking for favors, Asking for repetition, Requesting, Inviting						
bdslkorpus_ber_06	18-45	Medium	Offering, Talking about abilities, Expressing possibility, Talking about locations						
bdslskorpus_ber_07	18-45	Advanced	Emergency situations, Descriptions, Comparing things.						
bdslkorpus_ber_08	45-60	Advanced	Discussing Sensitive, Topics,						

			Accepting and Refusing, Supporting opinions						

Table 8. Corpus Data Structure

Transcript Flow for BdSL

	Translation	Lexeme/Sign	Mouth	Translation	Lexeme/Sign	Mouth	Moderator
00:00:00:00 00:00:07:41							
00:00:07:41 00:00:10:16							Did you experience anything sad, as well? Something like he experienced with his dad?
00:00:10:16 00:00:10:31				Something sad?			
00:00:10:31 00:00:10:36					$INDEX1*		
00:00:10:36 00:00:11:00							
00:00:11:00 00:00:11:07					SAD4*	traurig	
00:00:11:07 00:00:11:16							
00:00:11:16 00:00:11:18					I1		
00:00:11:18 00:00:11:21							
00:00:11:21 00:00:11:28							
00:00:11:28 00:00:11:36					$GEST-OFF^*		
00:00:11:36 00:00:11:39							
00:00:11:39 00:00:12:02				Well, actually I haven't experienced anything too sad so far.			
00:00:12:02 00:00:12:16					UNTIL-NOW3A		
00:00:12:16 00:00:12:23							
00:00:12:23 00:00:12:35					ACTUALLY1B	eigentlich	
00:00:12:35 00:00:13:02							
00:00:13:02 00:00:13:22					UNTIL-NOW3B	bis	
00:00:13:22 00:00:13:36							

00:00:13:36 00:00:13:48					NEVER- BEFORE1*	noch nie	
00:00:13:48 00:00:14:06							
00:00:14:06 00:00:14:15					SAD4*	traurig	
00:00:14:15 00:00:14:29							
00:00:14:29 00:00:14:43					$GEST-OFF^*		
00:00:14:43 00:00:14:46							

Table 9. Transcript Flow for BdSL

3.3.5 Training the Model for BdSLR

Assuming we have annotated dataset ready, we will start training models following different approaches. The reason because to follow different approaches to train the model is we don't know which one will give us close predictions.

Approach-1:

In the first approach, Detectron will be used which is Facebook AI Research's (FAIR) software system that implements state-of-the-art object detection algorithms, including Mask R-CNN. It is a high-performance codebase for object detection, covering bounding box and object instance segmentation outputs. It's written in Python and will be powered by the Caffe2 and PyTorch 1.0 deep learning framework. The goal of Detectron is to provide a high-quality, high-performance codebase for object detection research.

It is designed to be flexible in order to support rapid implementation and evaluation of novel research. Detectron includes implementations of the following object detection algorithms:

- Mask R-CNN
- RetinaNet
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN

And using the following backbone network architectures:

- ResNeXt{50,101,152}
- ResNet{50,101,152}
- Feature Pyramid Networks (with ResNet/ResNeXt)
- VGG16

Detectron 2 [10] is a next-generation open-source object detection system from Facebook AI Research. It's the updated version of Detectron. Detectron 2 addresses some legacy issues left in Detectron. As it is still in a transition mode, their models are not compatible: running inference with the same model weights will produce different results in the two code bases.

The major differences regarding inference are:

- The height and width of a box with corners (x1, y1) and (x2, y2) is now computed more naturally as width = x2 - x1 and height = y2 - y1; In Detectron, a "+ 1" was added both height and width.

- Note that the relevant ops in Caffe2 have adopted this change of convention with an extra option. So, it is still possible to run inference with a Detectron2-trained model in Caffe2. The change in height/width calculations most notably changes:
    - Encoding/decoding in bounding box regression.
    - Non-maximum suppression. The effect here is very negligible, though.

- RPN now uses simpler anchors with fewer quantization artifacts. In Detectron, the anchors were quantized and do not have accurate areas. In Detectron2, the anchors are center-aligned to feature grid points and not quantized.

- Classification layers have a different ordering of class labels. This involves any trainable parameter with shape (…, num_categories + 1, …). In Detectron2, integer labels [0, K-1] correspond to the K = num_categories object categories and the label "K" corresponds to the special "background" category. In Detectron, label "0" means background, and labels [1, K] correspond to the K categories.

- ROIAlign is implemented differently. The new implementation is available in Caffe2.

    - All the ROIs are shifted by half a pixel compared to Detectron in order to create better image-feature-map alignment. See layers/roi_align.py for details. To enable the old behavior, use ROIAlign(aligned=False), or POOLER_TYPE=ROIAlign instead of ROIAlignV2 (the default).

    - The ROIs are not required to have a minimum size of 1. This will lead to tiny differences in the output, but should be negligible.

- Mask inference function is different. In Detectron2, the "paste_mask" function is different and should be more accurate than in Detectron. This change can improve mask AP on COCO by ~0.5% absolute.

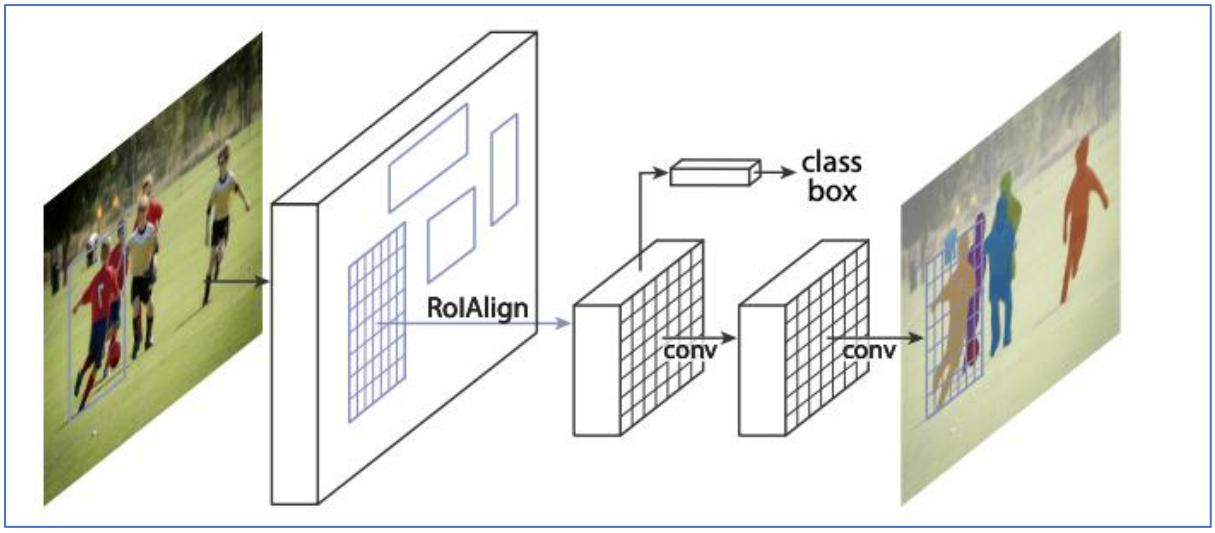


Figure 13. Detectron for segmentation

Faster R-CNN FPN architecture

The Base (Faster) R-CNN [11] with Feature Pyramid Network (Base-RCNN-FPN), which is the basic bounding box detector extendable to Mask R-CNN. Faster R-CNN detector with FPN backbone is a multi-scale detector that realizes high accuracy for detecting tiny to large objects, making itself the de-facto standard detector.

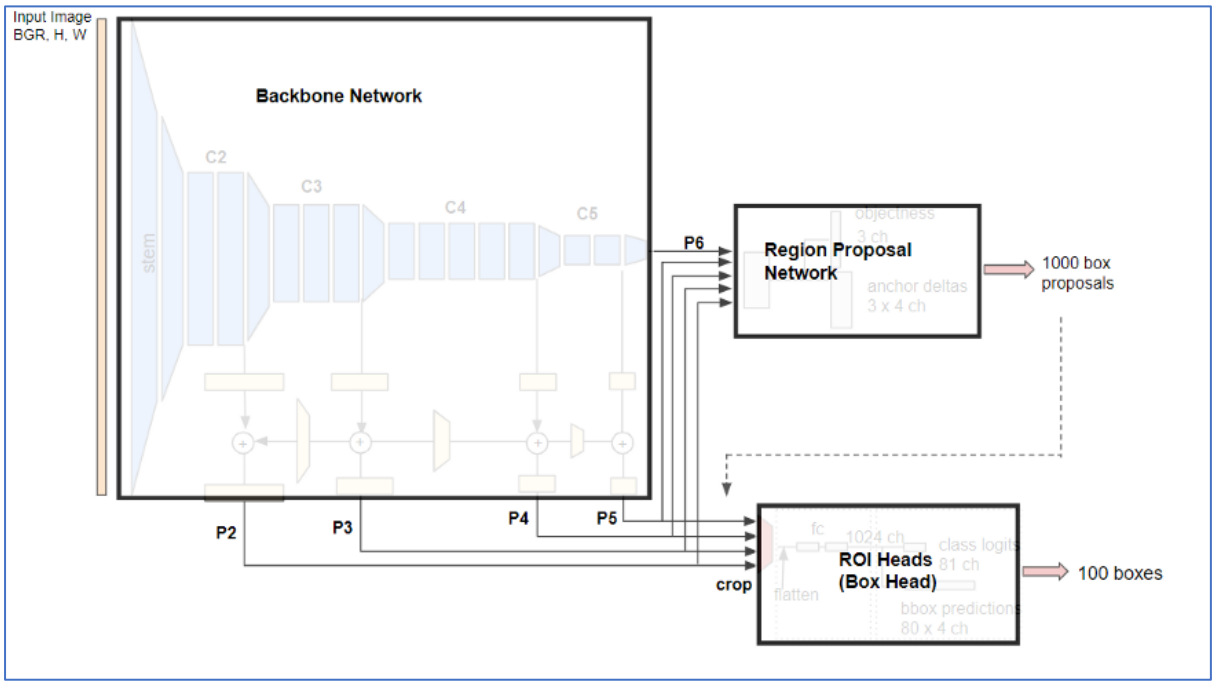


Figure 14. Meta architecture of Base RCNN FPN

The schematic above shows the meta-architecture of the network. Now there are three blocks in it, namely:

- Backbone Network: extracts feature maps from the input image at different scales. Base-RCNN-FPN's output features are called P2 (1/4 scale), P3 (1/8), P4 (1/16), P5 (1/32) and P6 (1/64). Note that non-FPN ('C4') architecture's output feature is only from the 1/16 scale.
- Region Proposal Network: detects object regions from the multi-scale features. 1000 box proposals (by default) with confidence scores are obtained.
- Box Head: crops and warps feature maps using proposal boxes into multiple fixed-size features, and obtains fine-tuned box locations and classification results via fully-connected layers. Finally 100 boxes (by default) in maximum are filtered out using non-maximum suppression (NMS). The box head is one of the sub-classes of ROI Heads. For example Mask R-CNN has more ROI heads such as a mask head.

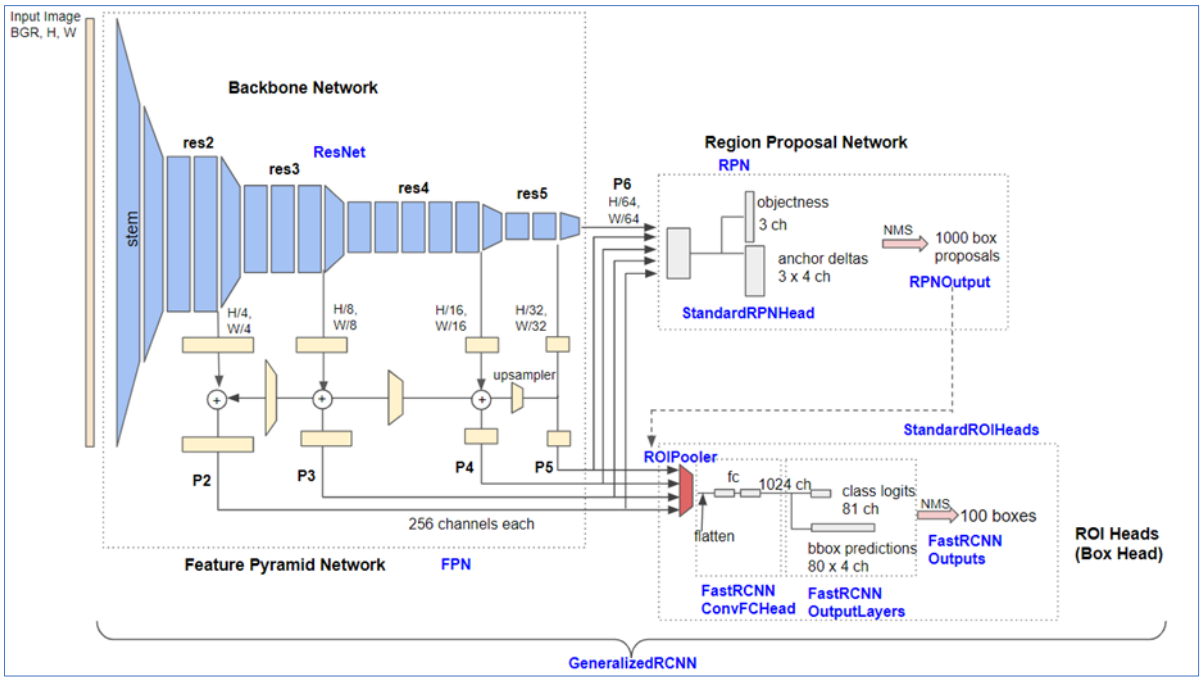


Figure 15. Detailed architecture of Base-RCNN-FPN. Blue labels represent class names

Structure of the detectron 2

The following is the directory tree of detectron2 (under the 'detectron2' directory). Please just look at the 'modeling' directory. The Base-RCNN-FPN architecture is built by the several classes under the directory.

```
detectron2
├─checkpoint  <- checkpointer and model catalog handlers
├─config      <- default configs and handlers
├─data        <- dataset handlers and data loaders
├─engine      <- predictor and trainer engines
├─evaluation  <- evaluator for each dataset
├─export      <- converter of detectron2 models to caffe2 (ONNX)
├─layers      <- custom layers e.g. deformable conv.
├─model_zoo   <- pre-trained model links and handler
├─modeling
│   ├─meta_arch <- meta architecture e.g. R-CNN, RetinaNet
│   ├─backbone  <- backbone network e.g. ResNet, FPN
│   ├─proposal_generator <- region proposal network
│   └─roi_heads <- head networks for pooled ROIs e.g. box, mask heads
├─solver       <- optimizer and scheduler builders
├─structures   <- structure classes e.g. Boxes, Instances, etc
└─utils        <- utility modules e.g. visualizer, logger, etc
```

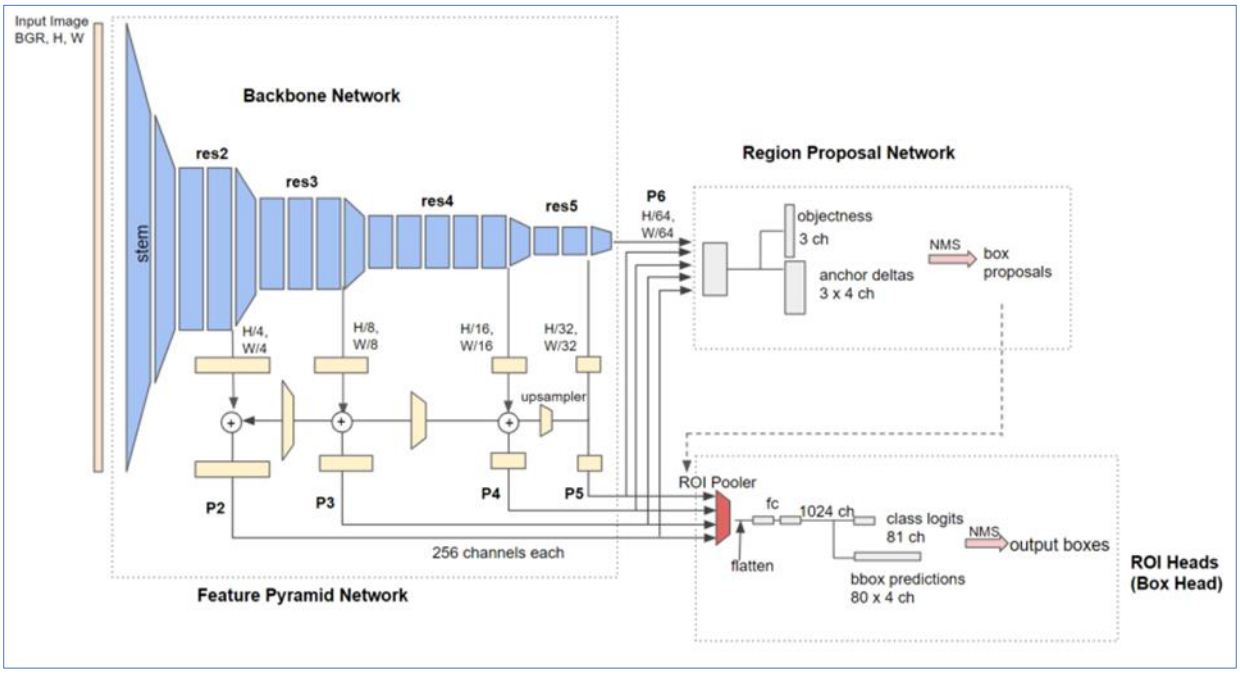


Figure 16. Detailed architecture of Base-RCNN-FPN (without class names)

Some benefits of Detectron & Detectron 2

- Generates segmentation mask for instances
- Simple to train as per the paper
- Utilizes a ResNet for pose estimation network
- Minimal domain knowledge

- Pipeline that can predict boxes, segments, and key points simultaneously

Cons:

- Not built immediately for pose estimation
- Semantic segmentation can lead to incorrect human bounding boxes and therefore incorrect joint poses
- Fails to account for rare poses and a few failure cases (i.e. overlapping objects)
- Fast, but not optimized for speed

BdSLR Model Training with Detectron

Firstly, as of plotted plan, we will annotate BdSLR dataset and follow the official approach of Detectron2 training. A fine tune will take place in this approach by replacing Detectron's dataset (COCO) by the BdSLR dataset. To optimize the image resolution, we will first resize images such that the short side will be around 600-800px and then run inference on the resized image. We will run the inference on our RTX 3080 Ti GPU.

The model will be an end-to-end trained Faster R-CNN using a ResNet-50-FPN backbone. A short training schedule will be used along with small input image size so that training and inference will be relatively fast. As a result, the box AP on COCO will be relatively low compared to our baselines.

Observations

According to the discussion of Detectron and Detectron2, there is a slim chance things might not perfectly fit with our demands, from server site point of view performance might be acceptable but for edge mobile device there might be slowness. We will be looking for bit faster and optimized algorithm which will be lightweight to run in mobile and web both platforms. And the model has to perform in real time. Detectron or Detectron2 just arrived based on new researches things might give trouble as they are not beaten tested yet.

Additionally, when someone performs sign languages, some positions might be complicated for estimating the pose. Sometimes Detectron and Detectron2 fails to understand rare and complex positions – yet not identified why.

Approach – 2

In the second approach, DensePose algorithm will be used which is a dense human pose estimation technique, designed to map all human pixels of an RGB image to a 3D surface-based representation of the human body. This is done through the introduction of a large-scale, manually annotated dataset, and a variant of Mask-RCNN, a simple, flexible framework for object instance segmentation.

DensePose establishes dense correspondences between RGB images and a surface-based representation of the human body. To do this AI researchers DensePose-COCO, a large-scale, ground-truth dataset with image-to-surface correspondences annotated on 50,000 COCO images.

Note: *The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.*

The DensePose-COCO [12] dataset was used to train DensePose-RCNN, a CNN-based system that delivers dense correspondences "in the wild", namely in the presence of complex backgrounds, occlusions, and scale variations.

Figure 17. 2D vs. 3D pose estimation

Figure 18. Visualization of Densepose

BdSLR dataset and Training with DensePose

A proposed end-to-end trained DensePose-RCNN model with a ResNet-101-FPN backbone from the official model zoo will be sued for experiment basis. BdSLR data set will be used replacing their zoo model dataset. The latest DensePose is using Detectron2 as backbone and the dataset preparation and training approach will be followed using approach 1.

Observations

Typically, Densepose is a model-based approach used to describe and infer human body poses and render 3D poses. It returns the 3D human mesh as poses. 3D pose estimation can be applied in very challenging tasks. On the other hand, there are still several challenges to overcome in 3D pose estimation: Model generalization, robustness to occlusion, and computation efficiency. DensePose is very resource hungry and the process is much slower in response. One of the biggest challenges is BdSLR will need 2D whereas DensePose provides 3D poses.

Approach – 3

The third and the final approach is to follow PoseNet along with other prominent tools and algorithm. PoseNet is a real-time pose detection technique with which body movement gesture geometry can be detected from human beings' poses in Image or Video. It works in both cases as single-mode (single human pose detection) and multi-pose detection (Multiple humans pose detection). More specifically, PoseNet is a deep learning TensorFlow model that allows us to estimate human pose by detecting body parts such as elbows, hips, wrists, knees, ankles, and form a skeleton structure of the pose by joining these points.

PoseNet is trained in MobileNet Architecture. MobileNet is a Convolutional neural network developed by google which is trained on the ImageNet dataset, majorly used for Image classification in categories and target estimation. It is a lightweight model which uses depth wise separable convolution to deepen the network and reduce parameters, computation cost, and increased accuracy.

PoseNet Architecture:

GoogLeNet architecture is used for developing a pose regression network. The original GoogLeNet architecture contains 22 layers that contain 6 Inception modules and two additional classifiers. Some changes had to be make in the architecture, these changes are:

- Replace every one of the three softmax classifiers with affine regressors. The softmax layers were taken out and every fully connected layer was modified to yield a pose vector of 7-dimensional representing position and orientation.
- Add another fully connected layer before the final regressors of feature size 2048. This was to form a localization feature vector which may then be explored for generalization.
- At test time we also normalize the quaternion orientation vector to unit length.

The various body joints detected by the pose estimation model are tabulated below:

Id	Part
0	nose
1	leftEye
2	rightEye
3	leftEar
4	rightEar
5	leftShoulder
6	rightShoulder
7	leftElbow
8	rightElbow
9	leftWrist
10	rightWrist
11	leftHip
12	rightHip
13	leftKnee
14	rightKnee
15	leftAnkle
16	rightAnkle

Table 10. Parameter Configuration

Running PosNet, all the positions of the each body part will be extracted from the given image/video gesture – this output will give new direction of calculating signs positions to take into measure.

FaceMesh [13] is a face geometry solution that estimates 468 3D face landmarks in real-time even on mobile devices. It employs machine learning (ML) to infer the 3D surface geometry, requiring only a single camera input without the need for a dedicated depth sensor. Utilizing lightweight model architectures together with GPU acceleration throughout the pipeline, the solution delivers real-time performance critical for live experiences.

Additionally, the solution is bundled with the Face Geometry module that bridges the gap between the face landmark estimation and useful real-time augmented reality (AR) applications. It establishes a metric 3D space and uses the face landmark screen positions to estimate face geometry within that space. The face geometry data consists of common 3D geometry primitives, including a face pose transformation matrix and a triangular face mesh. The pose estimations found from the PosNet will be used to generate accurate face geometry from the FaceMesh.

HandLandMarks: In HandLandMarks [14] the ability to perceive the shape and motion of hands can be a vital component in improving the user experience across different technological domains and platforms. It can form the basis for sign language understanding and hand gesture control, and can also enable the overlay of digital content and information on top of the physical world in augmented reality. PoseNet generated key points (pose estimation)

Mesh – PosNet keypoints for relations

DD-Net proposed Dense-Dialated Neural Network or DD-Net is an end-to-end deep neural network that mainly contains three parts. The first part is the backbone encoder-decoder structure for feature extraction and up-sampling. The second part is the dense dilated block (DDB), which has been integrated with the backbone. The last part is the computation of the loss. This network basically is for body/hand action recognition, implemented by keras tensorflow backend. It helps to skeleton-based action recognition from image.

VGG-16: VGG stands for Visual Geometry Group; The VGG model, or VGGNet that supports 16 layers is also referred to as VGG-16, which is a convolutional neural network model. The VGG-16 model achieves almost 92.7% top-5 test accuracy in ImageNet. ImageNet is a dataset consisting of more than 14 million images belonging to nearly 1000 classes. It replaces the large kernel-sized filters with several 3×3 kernel-sized filters one after the other, thereby making significant improvements over AlexNet.

As mentioned above, the VGGNet-16 supports 16 layers and can classify images into 1000 object categories, including keyboard, animals, pencil, mouse, etc. Additionally, the model has an image input size of 224-by-224.

LSTM [15] **optical flow and VGG-16 individual frame detection is required**

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997). LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically normal RNNs default behavior.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

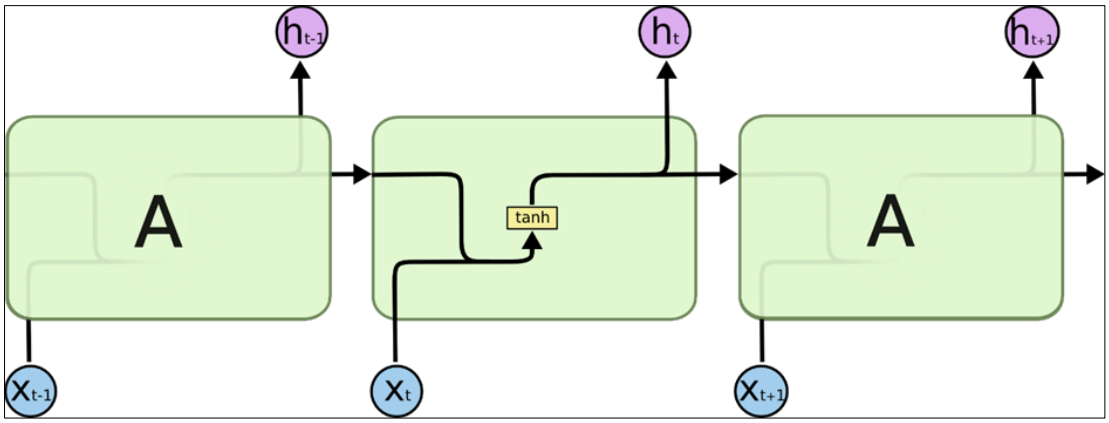


Figure 19. The repeating module in a standard RNN contains a single layer

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

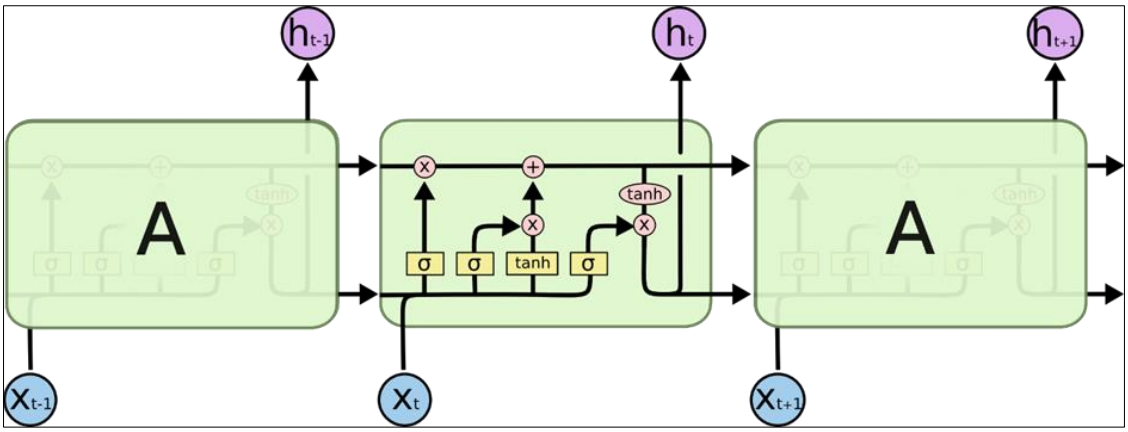


Figure 20. The repeating module in an LSTM contains four interacting layers

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

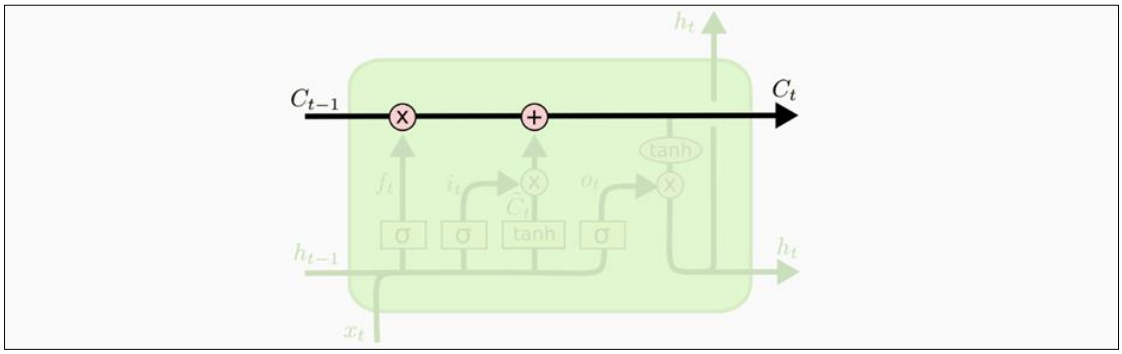


Figure 21. LSTM gate structure

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

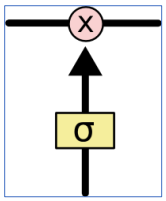


Figure 22. Gate layer

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through!"

An LSTM has three of these gates, to protect and control the cell state.

BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

BERT is based on the Transformer architecture. BERT is pre-trained on a large corpus of unlabeled text including the entire Wikipedia (that's 2,500 million words!) and Book Corpus (800 million words).

This pre-training step is half the magic behind BERT's success. This is because as we train a model on a large text corpus, our model starts to pick up the deeper and intimate understandings of how the language works. This knowledge is the Swiss army knife that is useful for almost any NLP task.

BERT is a "deeply bidirectional" model. Bidirectional means that BERT learns information from both the left and the right side of a token's context during the training phase.

The bidirectionality of a model is important for truly understanding the meaning of a language. Let's see an example to illustrate this. There are two sentences in this example and both of them involve the word "bank":

Will be trained in Bengali, missing words impose BERT, fully trained to push missing words.

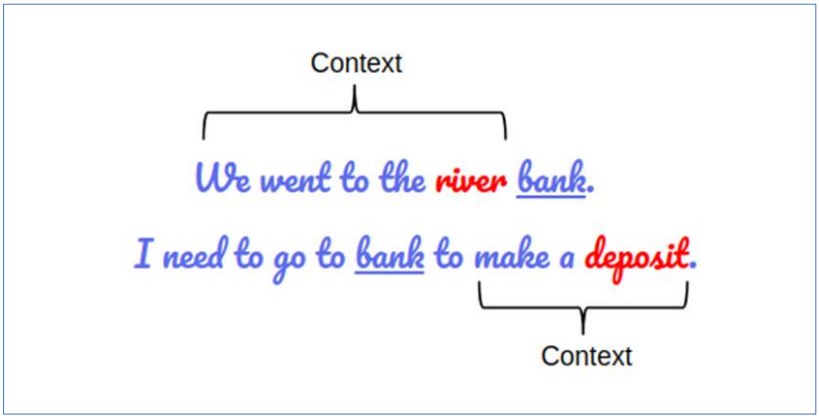


Figure 23. BERT captures both the left and right context

If we try to predict the nature of the word "bank" by only taking either the left or the right context, then we will be making an error in at least one of the two given examples.

One way to deal with this is to consider both the left and the right context before making a prediction. That's exactly what BERT does! We will see later in the article how this is achieved.

And finally, the most impressive aspect of BERT. We can fine-tune it by adding just a couple of additional output layers to create state-of-the-art models for a variety of NLP tasks.

From Word2Vec to BERT: NLP's Quest for Learning Language Representations [16]

"One of the biggest challenges in natural language processing is the shortage of training data. Because NLP is a diversified field with many distinct tasks, most task-specific datasets contain only a few thousand or a few hundred thousand human-labelled training examples." – Google AI.

Word2Vec and GloVe [16]

The quest for learning language representations by pre-training models on large unlabeled text data started from word embedding's like Word2Vec and GloVe. These embeddings changed the way we performed NLP tasks. We now had embeddings that could capture contextual relationships among words.

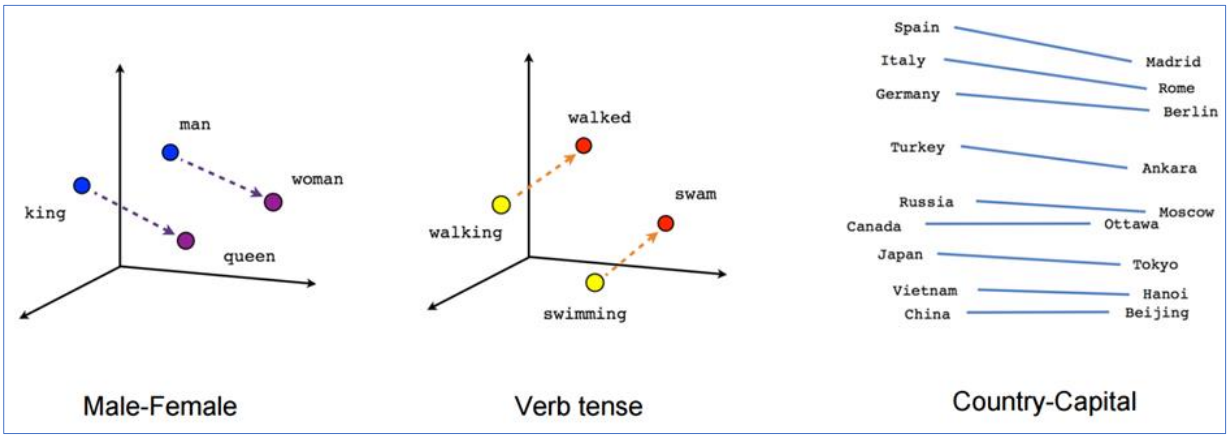


Figure 24. Word2Vec and GloVec

These embeddings were used to train models on downstream NLP tasks and make better predictions. This could be done even with less task-specific data by utilizing the additional information from the embeddings itself.

One limitation of these embeddings was the use of very shallow Language Models. This meant there was a limit to the amount of information they could capture and this motivated the use of deeper and more complex language models (layers of LSTMs and GRUs).

Another key limitation was that these models did not take the context of the word into account. Let's take the above "bank" example. The same word has different meanings in different contexts, right? However, an embedding like Word2Vec will give the same vector for "bank" in both the contexts.

That's valuable information we are losing.

Enter ELMO and ULMFiT [16]

ELMo was the NLP community's response to the problem of Polysemy – same words having different meanings based on their context. From training shallow feed-forward networks (Word2vec), we graduated to training word embeddings using layers of complex Bi-directional

LSTM architectures. This meant that the same word can have multiple ELMO embeddings based on the context it is in.

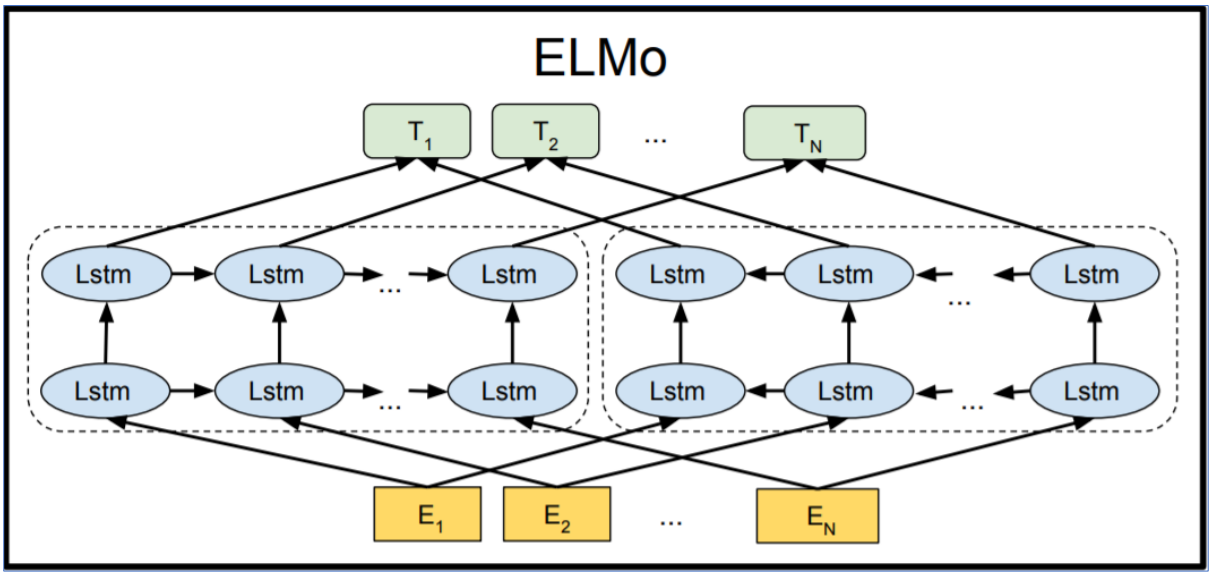


Figure 25. ELMo

That's when we started seeing the advantage of pre-training as a training mechanism for NLP.

ULMFiT took this a step further. This framework could train language models that could be fine-tuned to provide excellent results even with fewer data (less than 100 examples) on a variety of document classification tasks. It is safe to say that ULMFiT cracked the code to transfer learning in NLP.

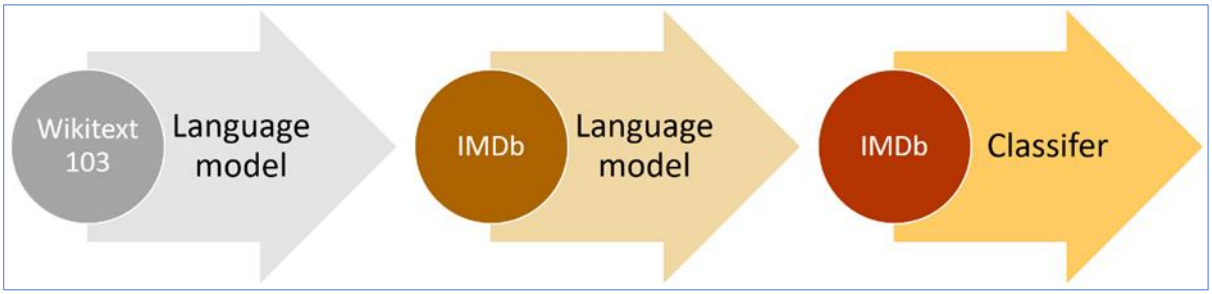


Figure 26. Transfer Learning in NLP

This is when we established the golden formula for transfer learning in NLP:

Transfer Learning in NLP = Pre-Training and Fine-Tuning

Most of the NLP breakthroughs that followed ULMFIT tweaked components of the above equation and gained state-of-the-art benchmarks.

A long way passed. Here is our Bangla-Bert! Bangla-Bert-Base is a pre-trained language model of Bengali language using mask language modeling described in BERT. [17]

Pre-train Corpus Details

Corpus was downloaded from two main sources:

- Bengali commoncrawl corpus downloaded from OSCAR [18]
- Bengali Wikipedia Dump Dataset [19]

After downloading these corpus, we preprocessed it as a Bert format which is one sentence per line and an extra newline for new documents.

Building Vocab

We used BNLP package for training Bengali sentence piece model with vocab size 102025. We preprocess the output vocab file as Bert format.

Training Details

- Bangla-Bert was trained with code provided in Google BERT's github repository [20]
- Currently released model follows bert-base-uncased model architecture (12-layer, 768-hidden, 12-heads, 110M parameters)
- Total Training Steps: 1 million
- The model was trained on a single Google Cloud TPU

Evaluation Results

LM Evaluation Results

```
global_step = 1000000
loss = 2.2406516
masked_lm_accuracy = 0.60641736
masked_lm_loss = 2.201459
next_sentence_accuracy = 0.98625
next_sentence_loss = 0.040997364
perplexity = numpy.exp(2.2406516) = 9.393331287442784
Loss for final step: 2.426227
```

Figure 27. After training 1 million steps here is the evaluation results

Downstream Task Evaluation Results

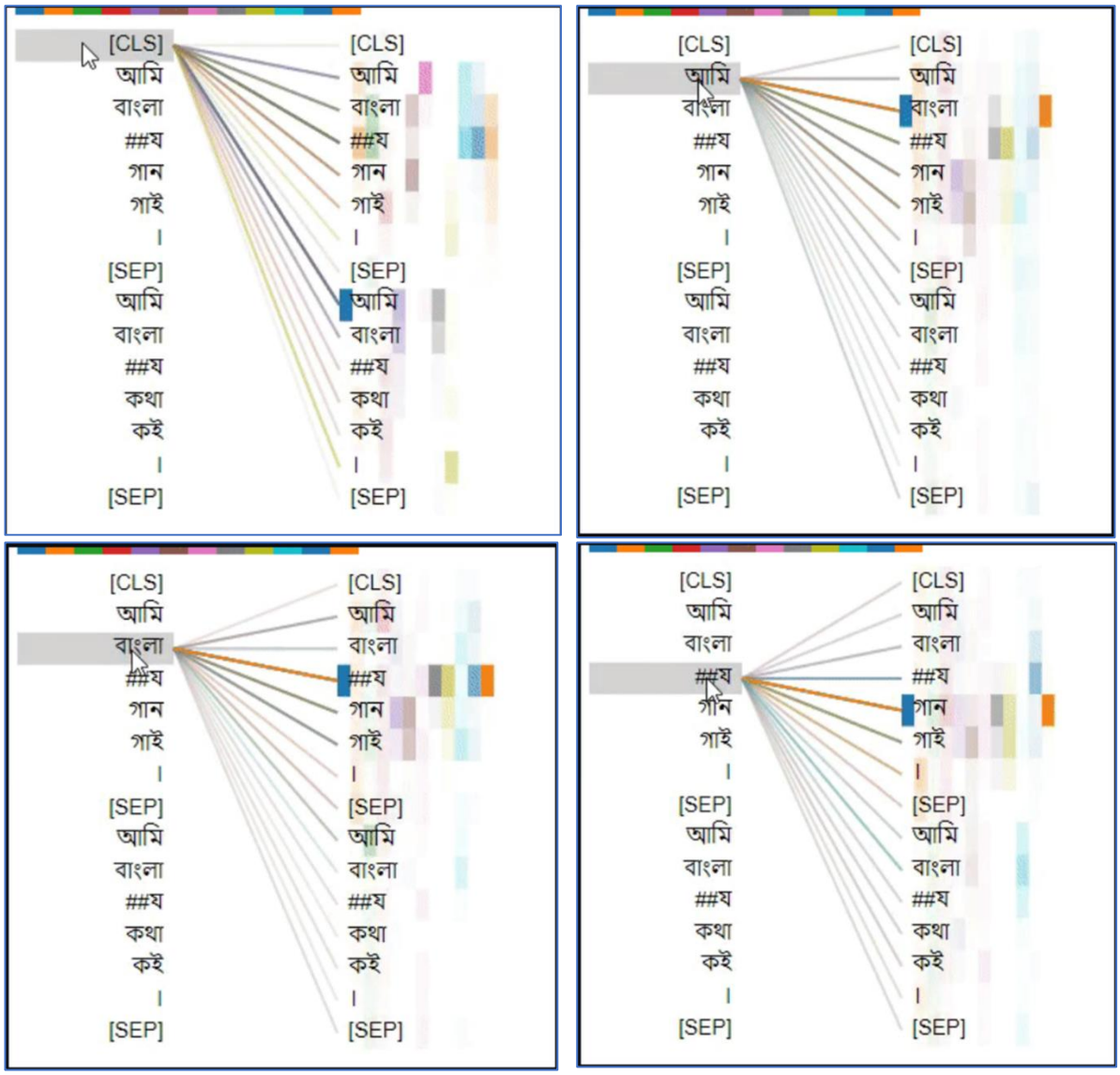
We evaluated Bangla-BERT-Base with Wikiann [21] Bengali NER datasets along with another benchmark three models (mBERT [22], XLM-R [23], Indic-BERT [24]).

Bangla-BERT-Base got a third-place where mBERT got first and XML-R got second place after training these models 5 epochs.

Base Pre-trained Model	F1 Score	Accuracy
mBERT-uncased	97.11	97.68
XLM-R	96.22	97.03
Indic-BERT	92.66	94.74
Bangla-BERT-Base	95.57	97.49

Table 11. F1 & Accuracy Difference

Bangla BERT Visualize



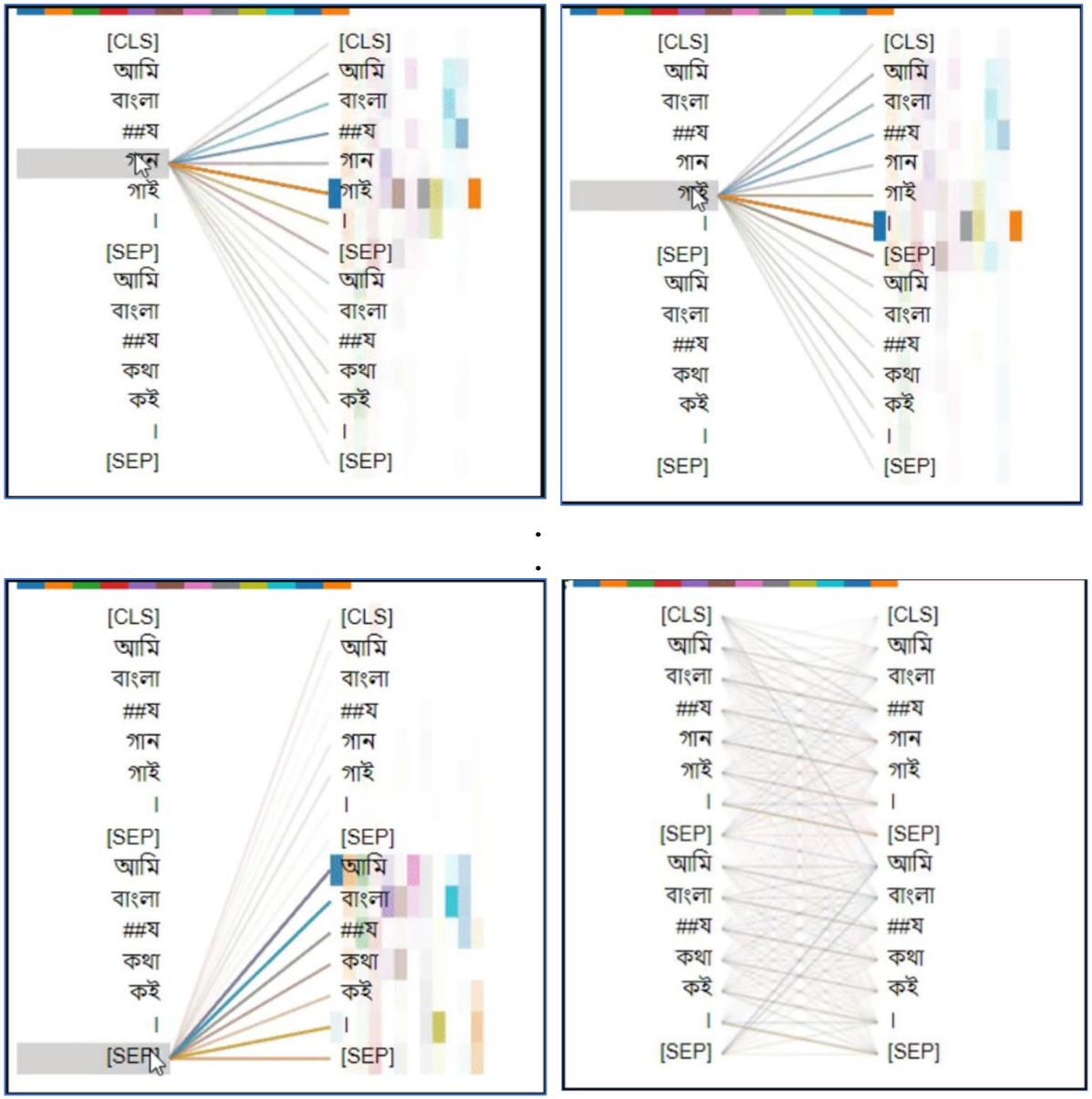


Figure 28. BERT Visualization

Generative Adversarial Networks (GANs)

Generative adversarial networks (GANs) are algorithmic architectures that use two neural networks, pitting one against the other (thus the "adversarial") in order to generate new, synthetic instances of data that can pass for real data. They are used widely in image generation, video generation and voice generation. One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not.

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.

GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

Conversion of Video to Keypoints/Sign

As of our understanding, the third approach that comprises a varied sets of model training of different parts of the gesture, expected to have close accuracy for the targeted goal, first we will train the mode to convert vide/images to generate keypoints of sings.

A deep learning model training is a process in which a learning algorithm is fed with sufficient training data to learn from. As we are using DL model in our SLR system, we need to train the model with enough data to identify and recognize the Bangla sign language. As soon as annotation is done, keypoints, coordinates and timeline are generated from BdSL dataset, which will be the input for model training phase.

In this phase for BdSL base model, with each image extracted from motion video (one second video can be break into 30 frame where per frame will be a single image), we will train some model architecture like **PoseNet**, **FaceMesh**, **HandLandMarks**, **DD-Net**, **VGG-16.** The training will be as in multichannel, which means all these model architecture will process one image by parallel and will get corresponding results. This training is an iterative process and by this process, models become trained via all the images. Roles of each training model described below:

- **PoseNet** helps to create a coordinate of hand and generate a sign label.
- **FaceMesh** will help to point face mesh and will generate a face sign label.
- **HandLandMarks** helps to generate hand sign label with coordinated positions.
- **DD-Net** helps to validate the hand motion whether it’s a part of sign movement or not.
- **VGG-16** ensures the object specification and detects sign performer.

Later on, from all the above corresponding result of trained model architecture, we will get a specific keypoints of feature or we can call it Feature Matrix. Employing LSTM on feature keypoints classifier will be generated from the specific matrix.

At this point, for the conversion process of keypoints to normal text, we have supposed constructed and trained 2 models which is Gloss to Text (**G2T**) tokenizer and Gloss to Word (**G2W**) tokenizer.

Conversion of Keypoints/Sign to Normal Text

Gloss to Word Tokenization

We will utilize spaCy (spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python.) and its Bengali-language model, and tell it that we will be doing our natural language processing using that model. Then we will assign our gloss string to text. Using nlp (text), we will process that text in Spacy

At this point, targeted text has already been tokenized, but **spaCy** stores tokenized text as a doc, and we would like to look at it in list form, so iterations through doc is needed, adding each word token it finds in the gloss string to a list called token_list ensuring all words are tokenized.

spaCy produces a list that contains each token as a separate item. Notice that it has recognized that contractions such as should not actually represent two distinct words, and it has thus broken them down into two distinct tokens. Bengalid dictionary will be there with nlp objects to crete documents with linguistic annotations and varisous NLP properties. After creating document, we are creating a token list.

In receipt of the keypoints or sign representation from **LSTM**, a mapping will be there with those keypoints into a gloss (word gloss). As we have our trained **G2T tokenizer** model, it will help to construct the sentence from word gloss. This sentences may consist of undefined form of grammar for which we will transform it to normal Bangla text sentence with the help of **POS Tagger (**POS Tagger is a process of grammatical tagging to a text to markup the unformed sentence to normal Bangla text).

It's obvious that for any conditions, if our **G2T tokenizer** model becomes incapable to generate sentence from the gloss (word gloss), then the model will guide the gloss to **G2W (Gloss to Word)** tokenizer model to handle it differently. This tokenizer will transform the word gloss to Bangla word. Later on, **BERT** model will process the word to sentence. Rest of the process will confirm the normal Bangla text with the help of **POS Tagger**.

Retraining model, fine tune based on output will be a repeating task until the satisfactory outcomes are there.

Observations - Third approach is bit instrumental – there are various kinds of algorithms and frameworks to consider. We are not yet sure that a close success rate is high but experimental results will give impression what is to consider and not.

3.3.6 System Architecture

High Level System Architecture

Bangla Sign Language Recognition and Conversion is the core module for our proposed application. There are several different stages of different activities. They can be divided into following groups:

- Data preparation from natural language
- Preparation of video data from text script
- Annotation process of video data
- Model Training (Training Dataset and Test Dataset)
- Predict output

The Bangla SLR system architecture is depicted into the below diagram (based 3rd Approach):

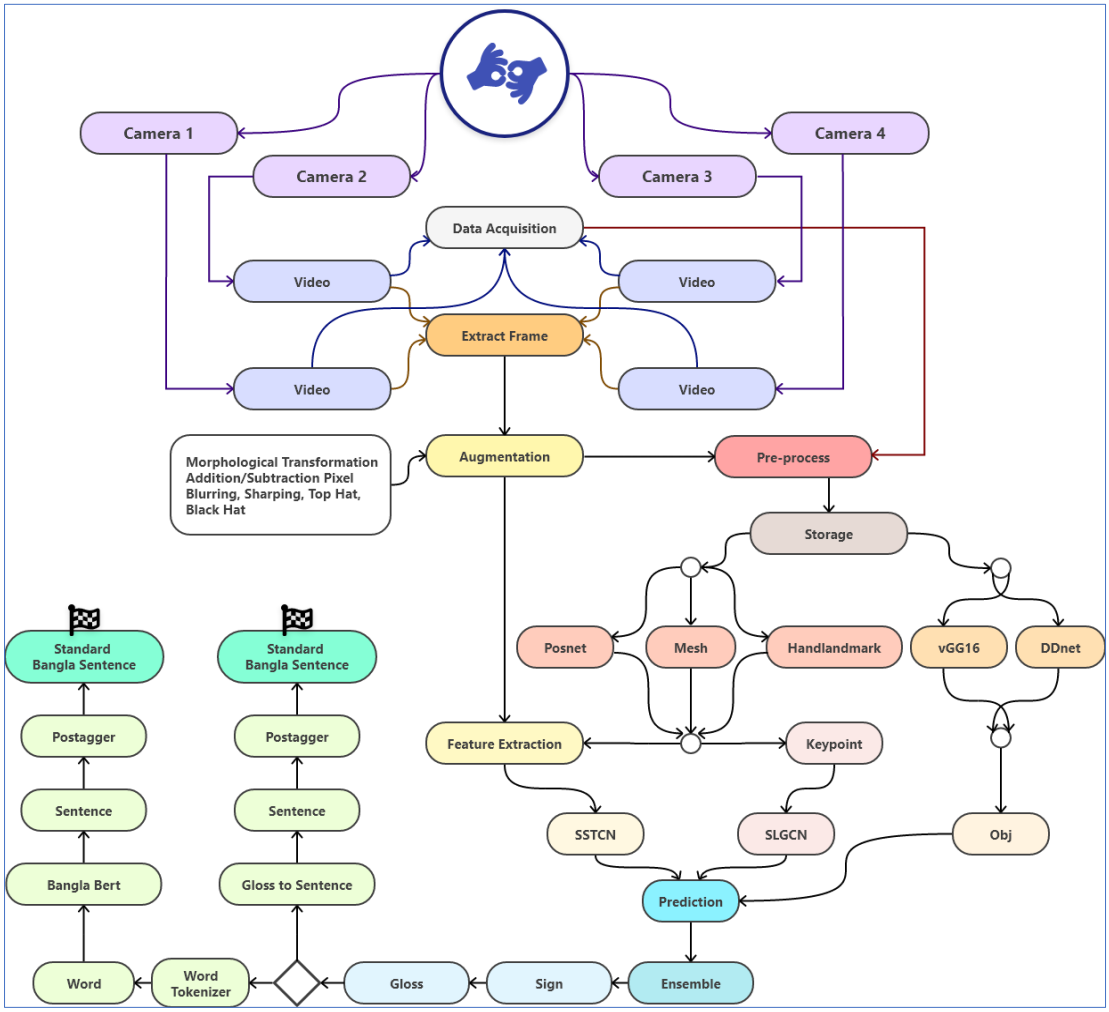


Figure 29. Bangla SLR Recognition and Conversion System Architecture

3.3.7 Model Test & Deploy

This is the most important part that we may go wrong and that is to properly evaluate our model, not to train the model on the entire dataset. A typical training/test runtime usually split into 70/30 ratio. We can test the model when 70% of the data has been collected from expected total data and trained to model. After that left 30% data would be specify for testing.

It's important to use new data when evaluating our model to prevent the likelihood of overfitting to the training set. However, sometimes it's useful to evaluate our model as we're building it to find that best parameters of a model - but we can't use the test set for this evaluation or else we'll end up selecting the parameters that perform best on the test data but maybe not the parameters that generalize best. To evaluate the model while still building and tuning the model, we create a third subset of the data known as the validation set. A typical train/test/validation split would be to use 60% of the data for training, 20% of the data for validation, and 20% of the data for testing.

For video data to sign gloss we use -

Classification metrics

When performing classification predictions, there's four types of outcomes that could occur.

- **True positives** are when you predict an observation belongs to a class and it actually does belong to that class.
- **True negatives** are when you predict an observation does not belong to a class and it actually does not belong to that class.
- **False positives** occur when you predict an observation belongs to a class when in reality it does not.
- **False negatives** occur when you predict an observation does not belong to a class when in fact it does.

These four outcomes are often plotted on a confusion matrix. The following confusion matrix is an example for the case of binary classification. You would generate this matrix after making predictions on your test data and then identifying each prediction as one of the four possible outcomes described above.

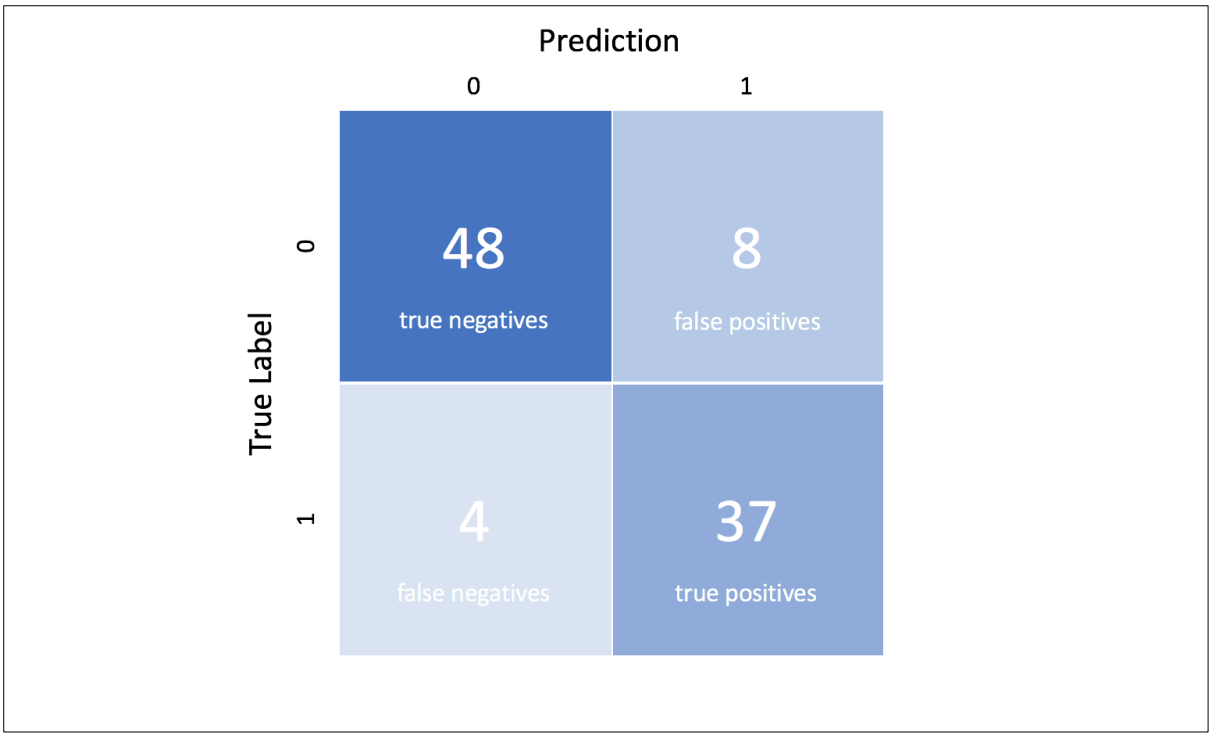


Figure 30. Classification Matrix

We can also extend this confusion matrix to plot multi-class classification predictions. The following is an example confusion matrix for classifying observations from the Iris flower dataset. The three main metrics used to evaluate a classification model are accuracy, precision, and recall.

FOR sign gloss to normal text as it's NLP related problem -

The ultimate goal of any machine learning model is to learn from examples and generalize some degree of knowledge regarding the task we're training it to perform. Some machine learning models provide the framework for generalization by suggesting the underlying structure of that knowledge. For example, a linear regression model imposes a framework to learn linear relationships between the information we feed it. However, sometimes we provide a model with too much pre-built structure that we limit the model's ability to learn from the examples - such as the case where we train a linear model on a exponential dataset. In this case, our model is biased by the pre-imposed structure and relationships.

Models with high bias pay little attention to the data presented; this is also known as underfitting.

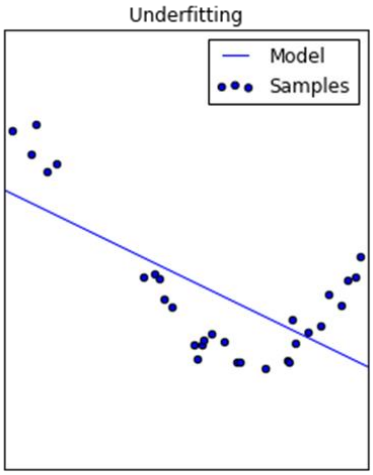


Figure 31. Underfitting

It's also possible to bias a model by trying to teach it to perform a task without presenting all of the necessary information. If you know the constraints of the model are not biasing the model's performance yet you're still observed signs of underfitting, it's likely that you are not using enough features to train the model.

On the other extreme, sometimes when we train our model it learns too much from the training data. That is, our model captures the noise in the data in addition to the signal. This can cause wild fluctuations in the model that does not represent the true trend; in this case, we say that the model has high variance. In this case, our model does not generalize well because it pays too much attention to the training data without consideration for generalizing to new data. In other words, we've overfit the model to the training data.

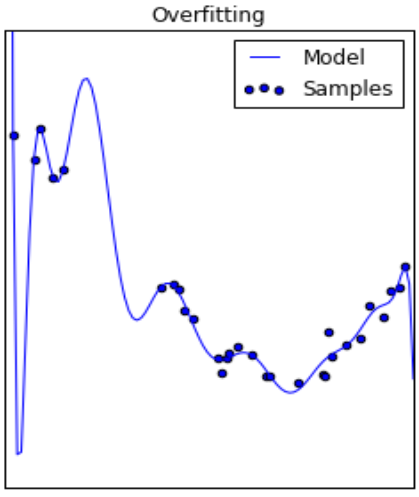


Figure 32. Overfitting

In summary, a model with high bias is limited from learning the true trend and underfits the data. A model with high variance learns too much from the training data and overfits the data. The best model sits somewhere in the middle of the two extremes.

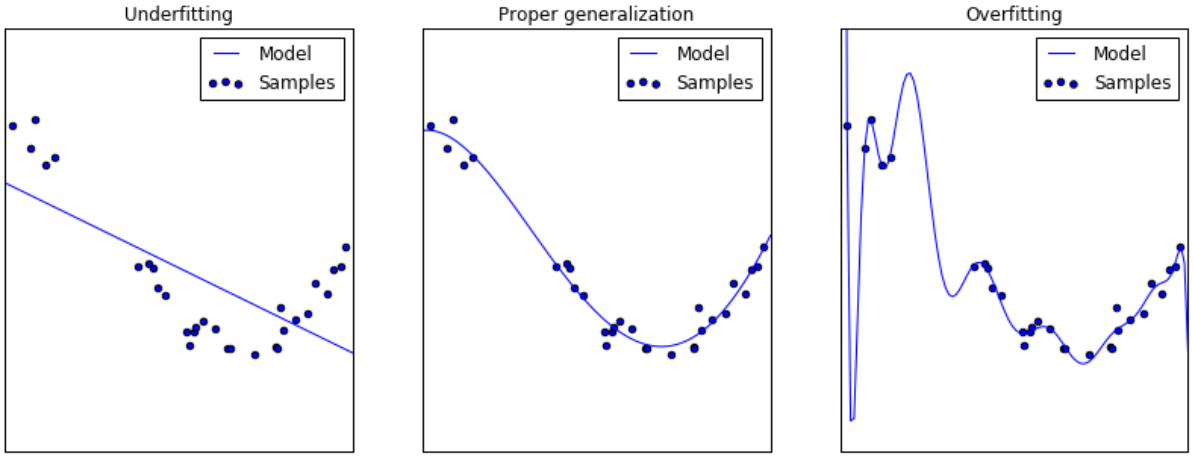


Figure 33. Understanding of Proper Generalization

Ethical Issues in Testing and Working with Deaf Communities

- Ensure test administrators are qualified to use sign language tests, e.g., level of sign proficiency, adequate training in sign language assessment and interpretation of test results, etc.
- Set and maintain high personal standards of competence in delivery of sign language assessments and interpretation of test results.
- Keep up to date about relevant changes and advances relating to test use and development. These may include changes in legislation and policy, which may impact on tests and test use, and the importance of updating test norms.
- Make sure that test materials (e.g., test instructions, items) and test data are kept securely at all times with limited access.
- Respect copyright law and agreements that exist with respect to a test, including any prohibitions on the copying or transmission of materials in electronic or other forms to other people, whether qualified or otherwise.
- Refrain from coaching individuals on actual test materials or other practice materials that might unfairly influence their test performance.
- Treat test results confidentially and securely in the case of online delivered tests.
- Make sure to obtain parental consent when testing children, and assent from children prior to testing. As part of the consent procedure, specify who will have access to results and define levels of confidentiality to individuals and their families before test administration.
- Obtain any relevant consent prior to releasing test results to others.
- Remove names and other personal identifiers from databases of results that are archived, for research use, development of norms or other statistical purposes.
- Produce a reasoned justification for the use of tests.
- Establish that the knowledge, skills, abilities, aptitudes or other characteristics, which the tests are intended to measure, are correlates of relevant behaviors in the context about which inferences are to be drawn.

- Determine that the test's technical and user documentation provides sufficient information to enable evaluation of the following:
    - Coverage and representativeness of test content, representative norm groups, difficulty level of content, etc.
    - Accuracy of measurement and reliability demonstrated with respect to relevant populations.
    - Validity (demonstrated with respect to relevant populations) and relevance for the required use.
    - Freedom from systematic bias in relation to the intended test taker groups.
    - Practicality, including time required, costs, and resource needs.
    - Avoid judging a test solely on the basis of face value, test-user testimonials, or advice from those with a vested commercial interest.

Issues Regarding Sign Language Test Development

In this section we address issues that are considered as important with particular regard to sign language tests.

Specific aspects for sign language test development

- Involve native signers at each stage of the process (from development to dissemination). These should ideally include Deaf native signers with academic training or other relevant training or experience. This is motivated by the fact that for some sign languages, limited research is available. Therefore, input is required from native signers to ensure that test items appropriately reflect the structure of the particular sign language.
- Involve Deaf/hearing people with different areas of expertise, such as linguists, psychologists, high-level interpreters, media design experts, computer programmers (for Web-/mobile-assisted sign language testing).
- Make sure that all materials are age appropriate for the test-takers and are also visually clear and accessible to the target population, avoid high memory load, do not require written responses, etc.
- Assure that the choice of elicitation techniques, item formats, test conventions, and procedures are readily accessible to all intended populations (e.g., children, adults).
- Ascertain that item content and stimulus materials are familiar to all intended populations (e.g., images that are suitable for children might not be appropriate to be used with adults).
- The language used in the directions and items themselves as well as in the handbook/manual should be appropriate for all cultural and language populations for whom the test or instrument is intended.
- In case of a test adaptation, test developers/publishers should ensure that the adaptation process takes full account of linguistic and cultural differences among the populations for whom adapted versions of the test or instrument are intended. (See also Selected references for more information regarding sign language test adaptation.)

Testing Deaf Children in Different Contexts

General Issues Testing Deaf Children

- When carrying out assessments, it is important to use different approaches to obtain as much information as possible about the child's language skills. This can be achieved using a range of methods
    - Include both receptive and productive instruments to assess sign language.
    - Include different instruments whenever possible that focus on different aspects of sign languages, such as vocabulary, grammar, narrative skills.
    - Environment (carry out assessments in a range of settings, e.g., clinical, school and/or home settings). The child is likely to be most relaxed in a familiar setting and will most likely respond differently depending on the test environment.
    - It is important to realize that all testing will be "inauthentic", however we can try to achieve the closest to natural settings.
    - Try to use more than one sign language test, if available. Different tests may yield different results or test slightly different aspects of a child's language skills.

- Appropriate standards for assessors
    - High level of sign language skills, preferably fluent in the given sign language.
    - The ability to communicate flexibly to meet the needs of individual Deaf children (many Deaf children today have exposure to a sign language (e.g., BSL)/ sign systems (e.g., Signed English) skills) / sign accompanying spoken language (e.g., Sign Supported English).
    - Experience in working with Deaf children.
    - People involved in the assessment process (teamwork): We stress the importance of Deaf and hearing teamwork when carrying out language assessments as each member of the team will have a different set of skills.

- Aims of assessment
    - The aims of the assessment need to be clear.
    - It is important that the assessment is actually testing what you want to test, e.g., vocabulary knowledge vs. reading skills (validity).

- Interaction
    - It is vital to include live interaction with the child you are testing. Computer-based assessments are becoming more and more popular due to its efficiency, but it does not always capture the child's ability to use language interactively.

- General data protection
    - Ensure that the data collected is secure and not identifiable.
    - Describe what happens to the data when the project ends or if the person responsible for the data leaves.
    - Follow data protection laws of your country and keep up to date with changes in law in your country.

- Ensure that appropriate ethical approval is obtained before collecting and storing personal data.
- Ensure that individuals and/or their parent/guardian consents to collected data about them being stored and to any specific use being made of their personal data including test scores.

- Responsibility
  - The people who carry out sign language assessments are usually the first people to get a good understanding of the child being assessed. Assessors have a responsibility to ensure the outcomes of assessments are comprehensible and can be shared with the people who work with the child on a daily basis.

Deploy /Host Trained Model

Hosting Infrastructure

There are lots of cloud services now a days in the AI platform (aka AISaaS or MLSaaS), AWS, Microsoft Azure and so on. Everyone knows, the cloud is the least expensive way to host AI development and production, but it turns out that the best solution may depend on where we are on our AI journey, how intensively we will be building out our AI capabilities, and what our end-game looks like.

Cloud computing is attractive for AI

Cloud service providers (CSPs) have extensive portfolios of development tools and pre-trained deep neural networks for voice, text, image, and translation processing. Much of this work stems from the internal development of AI for in-house applications, so it is pretty robust. Microsoft Azure for example, offers around 30 pre-trained networks and tools which can be accessed by your cloud-hosted application as APIs. Many models can even be customized with users' own data, such as specific vocabulary or images. Amazon SageMaker provides cradle-to-production AI development tools, and AWS offers easy chatbot, image recognition, and translation extensions to AWS-hosted applications. Google has a pretty amazing slew of tools as well. Most notably, perhaps, is its AutoML, which builds **Deep Learning** neural networks auto-magically, saving weeks or months of labor in some cases.

All of these tools have a few things in common. First, they make building AI applications seem enticingly easy. Since most companies struggle to find the right skills to staff an AI project, this is very attractive. Second, they offer ease of use, promising click-and-go simplicity in a field full of relatively obscure technology. Lastly, all these services have a catch—for the most part, they require that the application you develop in their cloud runs in their cloud.

And hence, these services are therefore tremendously sticky. If you use Amazon Poly to develop chatbots, for example, you can never move that app off of AWS. If you use Microsoft's pre-trained DNNs for image processing, you cannot easily run the resulting application on your own servers. You will probably never see a Google TPU in a non-Google data center, nor be able to use the Google AutoML tool if you later decide to self-host the development process.

Being stickiness is not necessarily a bad thing. After all, elastic cloud services can offer a flexible hardware infrastructure for AI, complete with state-of-the-art GPUs or FPGAs to accelerate the training process and handle the flood of inference processing you hope to attract to your new AI (where the trained neural network is actually used for real work or play). You don't have to deal with complex hardware configuration and purchase decisions, and the AI software stacks and development frameworks are all ready-to-go. For these reasons, many AI startups begin their development work in the cloud, and then move to their own infrastructure for production.

Cloud Computing is Expensive

A lot of AI development, especially training deep neural networks, eventually demands massive computation in large scale. Furthermore, based on the learning nature, we can't stop training a (useful) network; we always need to keep it fresh with new data and features, or perhaps build a completely new network to improve accuracy using new algorithms as they come out. The publicly available research says that this level of compute can become pretty **expensive in the cloud, costing 2-3 times as much as building our own private cloud to train and run the neural nets**. Note that one can reserve dedicated GPUs for longer periods in the cloud, significantly lowering the costs, but owning own hardware remains the lowest cost for ongoing, GPU-heavy workloads.

Reasons for cloud or self-hosting

Setup: Starting an AI Project can take a lot of time, effort, and expenses. Cloud AI services can greatly reduce the pains of getting started, and some hardware vendors offer bundles of hardware and software to even the playing field with the CSPs. Dell EMC's "AI Ready Solutions" for Deep and Machine Learning, for example, comes complete with GPUs and integrated software stacks, all designed to smooth the on-ramp to building AIs.

Security: Some industries are tightly regulated and require on-premises infrastructure. Others, such as financial services, deem it too risky to put sensitive information into a cloud.

Data gravity: This is the most important factor for some organizations. Simply, if our data is in the cloud, we should build our AIs and put our apps there as well. If our data is on-premises, the hassle and costs of data transfers can be onerous, especially considering the massive size of neural network training dataset. For that reason, it usually makes sense to build your AI on-prem as well.

For production deployment trained ML/DL model need to be uploaded to the cloud or private cloud, so that we can send prediction requests to the model. In order to deploy our trained model on AI Platform, we must save our trained model using the tools provided by machine learning framework. This involves serializing the information that represents our trained model into a file which we can deploy for prediction in the cloud/private cloud. Then we upload the saved model to a Cloud Storage bucket, and create a model resource on AI Platform pointing saved model to the clouds, this ensures that the uploaded file the saved file that we prepared earlier.

3.4. Approach for Text to Sign Puppet (T2SP)

3.4.1 Text to Sign Puppet Conversion

For extended/bidirectional communication between normal people and the deaf person this system (text to sign puppet) will enable another communication channel using which normal people can to talk to deaf people. As mentioned earlier – this topic is also in the list of researches and development and no implementation done yet for Bangla text to Puppet translation.

As SLR methodologies have discussed and the ways of preparing BdSLR dataset has been covered, this SLR dataset will be cast-off here for the conversion and translation process of text to sign puppet. Team explored some extensive libraries, selected one of many **Spacy (spaCy)**. This library feeds some pre-trained model, platform like NLTK and process like NER to make the conversion successful.

Spacy:

Spacy is an open-source software library for advanced natural language processing, written in the programming languages Python [25]

Unlike Natural Language Toolkit (NLTK) which is widely used for teaching and research, Spacy focuses on providing software for production usage. Spacy also supports deep learning workflows that allow connecting statistical models trained by popular machine learning libraries like **TensorFlow**, **PyTorch** or **MXNet** through its own machine learning library Thinc. Using Thinc as its backend, Spacy features convolutional neural network models for part-of-speech tagging, dependency parsing, text categorization and Named Entity Recognition (NER).

Spacy Features: The features that we need in model training of T2SP-

- Non-destructive tokenization.
- Built-in support for trainable pipeline components such as Named Entity Recognition, Part-of-speech tagging, dependency parsing, Text classification, Entity Linking and more.
- Multi-task learning with pre-trained transformers like BERT.
- Support for custom models in **PyTorch**, **TensorFlow** and other frameworks.
- State-of-the-art speed and accuracy.
- Production-ready training system.
- Built-in visualizers for syntax and named entities.
- Easy model packaging, deployment and workflow management.

Library Architecture [25]**:** The central data structures in Spacy are the **Language** class (Processing class that turns text into Doc objects. Different languages implement their own subclasses of it. The variable is typically called nlp), the Vocab and the Doc object (A container for accessing linguistic annotations.). The Language class is used to process a text and turn it

into a Doc object. It's typically stored as a variable called nlp. The Doc object owns the sequence of tokens and all their annotations. By centralizing strings, word vectors and lexical attributes in the Vocab, we avoid storing multiple copies of this data. This saves memory, and ensures there's a single source of truth.

Text annotations are also designed to allow a single source of truth: the Doc object owns the data, and Span (A slice from a Doc object) and Token (An individual token, i.e. a word, punctuation symbol, whitespace, etc.) are views that point into it. The Doc object is constructed by the Tokenizer, and then modified in place by the components of the pipeline. The Language object coordinates these components. It takes raw text and sends it through the pipeline, returning an annotated document. It also orchestrates training and serialization.

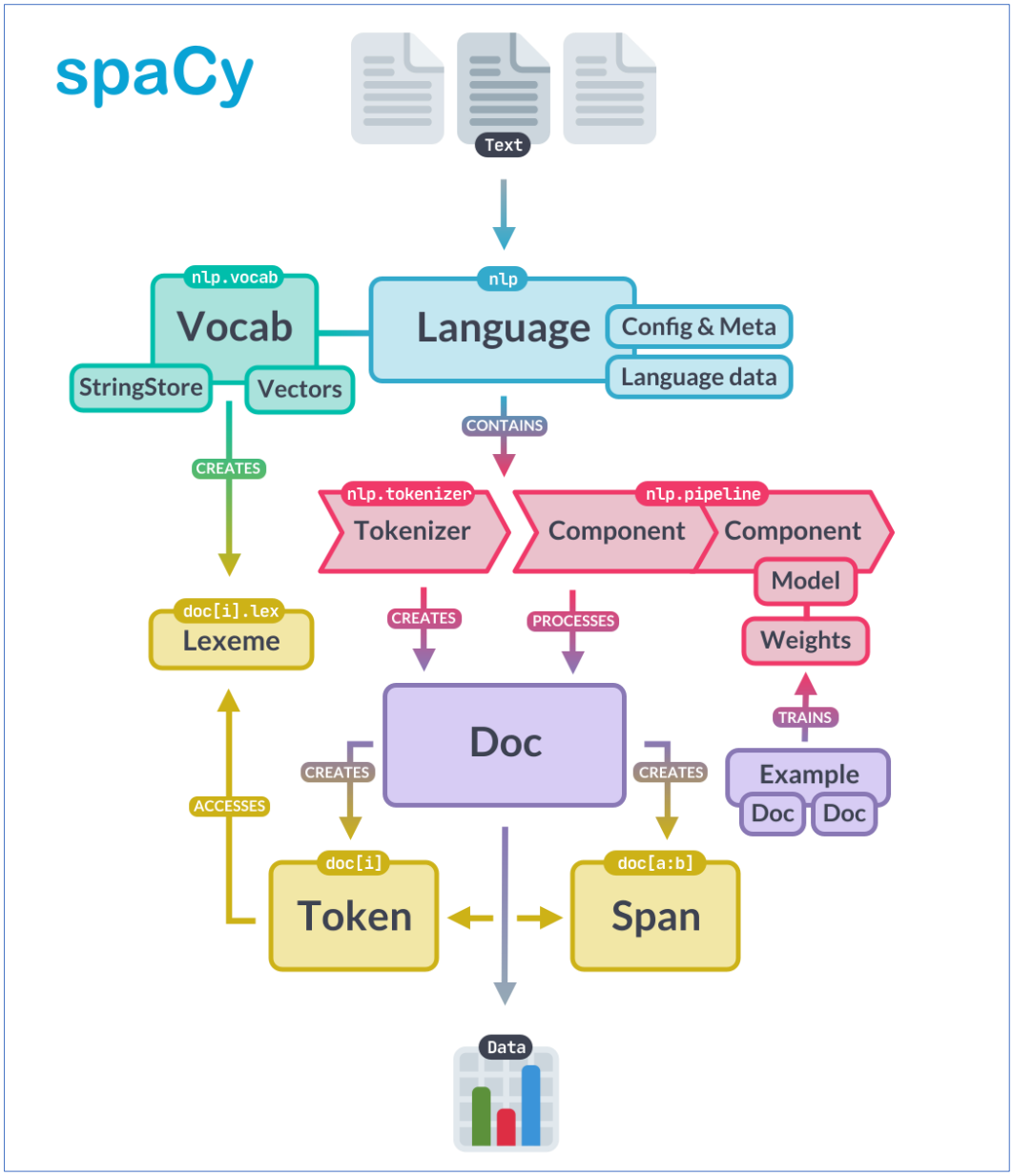


Figure 34. Spacy Library Architecture

Processing pipeline: The processing pipeline consists of one or more pipeline components that are called on the Doc in order. The tokenizer runs before the components. Pipeline components can be added using **Language.add_pipe**. They can contain a statistical model and trained weights, or only make rule-based modifications to the Doc. Spacy provides a range of built-in components for different language processing tasks and also allows adding custom components.

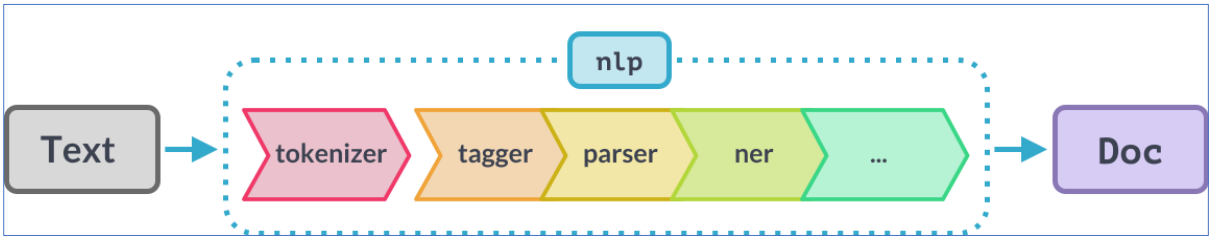


Figure 35. Spacy Processing Pipeline

Spacy Model

Package naming conventions: In general, Spacy expects all pipeline packages to follow the naming convention of **[lang]_[name].** For Spacy's pipelines, we also chose to divide the name into three components:

- Type: Capabilities (e.g. core for general-purpose pipeline with tagging, parsing, lemmatization and named entity recognition, or dep for only tagging, parsing and lemmatization).
- Genre: Type of text the pipeline is trained on, e.g. web or news.
- Size: Package size indicator, sm, md, lg or trf (sm: no word vectors, md: reduced word vector table with 20k unique vectors for ~500k words, lg: large word vector table with ~500k entries, trf: transformer pipeline without static word vectors)

Trained pipeline design: The Spacy trained pipelines are designed to be efficient and configurable. For example, multiple components can share a common "token-to-vector" model and it's easy to swap out or disable the lemmatizer. The pipelines are designed to be efficient in terms of speed and size and work well when the pipeline is run in full.

When modifying a trained pipeline, it's important to understand how the components depend on each other. The **tagger**, **parser** and **ner** components were all independent components depend on earlier components in the pipeline. As a result, disabling or reordering components can affect the annotation quality or lead to warnings and errors.

- The Tok2Vec component may be a separate, shared component. A component like a tagger or parser can listen to an earlier tok2vec or transformer rather than having its own separate tok2vec layer.
- Rule-based exceptions move from individual components to the **attribute_ruler**. Lemma and POS exceptions move from the tokenizer exceptions to the attribute ruler and the tag map and morph rules move from the tagger to the attribute ruler.

- The lemmatizer tables and processing move from the vocab and tagger to a separate lemmatizer component.

CNN/CPU Trained pipeline design

In the small/medium/large models:

- The tagger, morphologizer and parser components listen to the tok2vec component.
- The **attribute_ruler** maps **token.tag** to **token.pos** if there is no **morphologizer**. The **attribute_ruler** additionally makes sure whitespace is tagged consistently and copies **token.pos** to **token.tag** if there is no tagger. For English, the attribute ruler can improve its mapping from token.tag to token.pos if dependency parses from a parser are present, but the parser is not required.
- The **lemmatizer** component for many languages (Catalan, Dutch, English, French, Greek, Italian Macedonian, Norwegian, Polish and Spanish) requires **token.pos** annotation from either **tagger+attribute_ruler** or morphologizer.
- The **ner** component is independent with its own internal tok2vec layer.

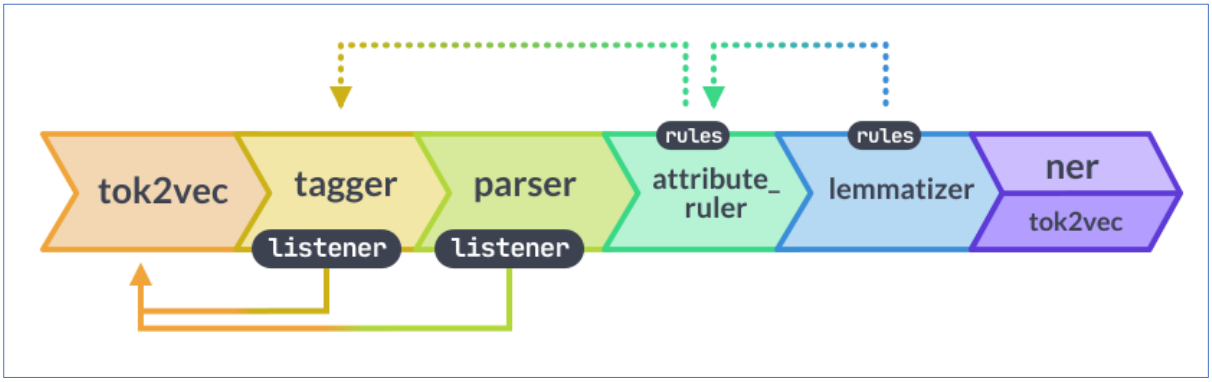


Figure 36. CNN/CPU Trained pipeline

Transformer pipeline design

In the transformer (**trf**) models, the **tagger**, **parser** and **ner** (if present) all listen to the **transformer** component. The **attribute_ruler** and **lemmatizer** have the same configuration as in the CNN models.

At the moment there is no prepared Bangla data corpus. We have developed a short corpus and will train that corpus through CNN/CPU pipeline. After getting an expected result for that specific corpus and developing the large data corpus, we will train that corpus through Transformer/GPU pipeline.

Pre-trained Model

Pre-trained model is those model which is already trained by other developers and with different high quality datasets which all has acceptable by others. We will obtain all those necessary pre-trained models that needs for our training from Spacy [26] and NLTK [27].

Natural Language Toolkit (NLTK)

Natural Language Processing (NLP) is a process of manipulating or understanding the text or speech by any software or machine. An analogy is that humans interact and understand each other's views and respond with the appropriate answer. In NLP, this interaction, understanding, and response are made by a computer instead of a human.

NLTK (Natural Language Toolkit) Library is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.

Learning Natural Language Toolkit will help us add an extra skill and also enhance our knowledge of NLP. Learning NLTK library is also beneficial to enhance any workings in AI and Natural Language Processing with Python.

Modules: The toolkit is implemented as a collection of independent modules, each of which defines a specific data structure or task. A set of core modules defines basic data types and processing systems that are used throughout the toolkit. The token module provides basic classes for processing individual elements of text, such as words or sentences. The tree module defines data structures for representing tree structures over text, such as syntax trees and morphological trees. The probability module implements classes that encode frequency distributions and probability distributions, including a variety of statistical smoothing techniques. The remaining modules define data structures and interfaces for performing specific NLP tasks.

Parsing Modules: The parser module defines a high-level interface for producing trees that represent the structures of texts. The **chunkparser** module defines a sub-interface for parsers that identify non-overlapping linguistic groups (such as base noun phrases) in unrestricted text.

Four modules provide implementations for these abstract interfaces. The **srparser** module implements a simple shift-reduce parser. The **chartparser** module defines a flexible parser that uses a chart to record hypotheses about syntactic constituents. The **pcfgparser** module provides a variety of different parsers for probabilistic grammars. And the **rechunkparser** module defines a transformational regular expression based implementation of the chunk parser interface.

Tagging Modules: The tagger module defines a standard interface for augmenting each token of a text with supplementary information, such as its part of speech or its WordNet synset tag; and provides several different implementations for this interface.

Visualization: Visualization modules define graphical interfaces for viewing and manipulating data structures, and graphical tools for experimenting with NLP tasks. The **draw.tree** module provides a simple graphical interface for displaying tree structures. The **draw.tree** edit module provides an interface for building and modifying tree structures.

The **draw.plot** graph module can be used to graph mathematical functions. The **draw.fsa** module provides a graphical tool for displaying and simulating finite state automata. The **draw.chart** module provides an interactive graphical tool for experimenting with chart parsers. The visualization modules provide interfaces for interaction and experimentation; they do not directly implement NLP data structures or tasks. Simplicity of implementation is therefore less of an issue for the visualization modules than it is for the rest of the toolkit.

Text Classification: The **classifier** module defines a standard interface for classifying texts into categories. This interface is currently implemented by two modules. The **classifier.naivebayes** module defines a text classifier based on the Naive Bayes assumption. The **classifier.maxent** module defines the maximum entropy model for text classification, and implements two algorithms for training the model: Generalized Iterative Scaling and Improved Iterative Scaling. The **classifier.feature** module provides a standard encoding for the information that is used to make decisions for a particular classification task. This standard encoding allows students to experiment with the differences between different text classification algorithms, using identical feature sets. The **classifier.featureselection** module defines a standard interface for choosing which features are relevant for a particular classification task. Good feature selection can significantly improve classification performance.

Named Entity Recognition (NER)

NER delves with identification and classification of named entities (NEs) in a given sentence, such as person, organization, place, time, quantities, and monetary values. NER plays a key role in larger information extraction (IE) from natural language texts. NER finds practical use in a wide range of applications such as machine translation, speech recognition, chat bots, and search engines.

Traditional approaches to NER include matching a list of names with documents to identify NEs and other simple rule-based approaches. The Conference on Natural Language Learning (CoNLL) conducted a shared task in 2003, which saw a lot of significant contributions to NER. Many statistical NLP approaches were presented here that achieved success in NER with groundbreaking results. Some of these approaches are discussed in the following sections.

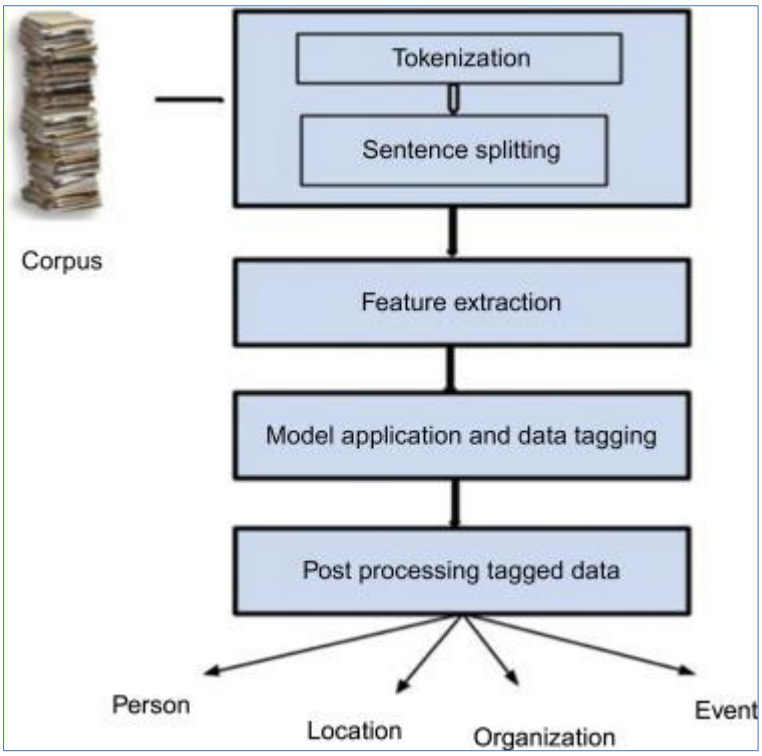


Figure 37. Named Entity Recognition

NER pipeline: A typical NER system pipeline includes preprocessing the data such as tokenization, sentence splitting, feature extraction, applying ML models on the data for tagging, and then post-processing to remove some tagging inconsistencies.

Statistical NLP methods: NER can be treated as a standard classification problem, for example, identifying if a token is an entity or not. Statistical machine learning algorithms can be applied to solve this problem. Supervised classifiers yielded better NER systems compared to rule-based ones, as they are easier to maintain and adapt to different domains. Hidden Markov models (HMMs) and support vector machine (SVM)-based models were used in NER tasks with varying degree of success.

Features for NER: Most of the systems employed for solving NER use a large set of features. Domain adaptation of NER entails feature engineering, that is, addition or deletion of features for that particular data. Feature engineering involves manually constructing features for a particular domain, which involves understanding of the domain and also of the classifier used. The effect of each feature can be manually measured by simple inclusion and exclusion of features. There are a number of feature selection algorithms used in machine learning. Methods such as adding features based on domain knowledge, forward selection (i.e., adding one feature at a time to evaluate performance), and backward selection (i.e., removing one feature at a time to evaluate performance), can be used initially. Models based on simple Information Gain and chi-square help improve the feature selection process drastically.

Stemmers

Stemmer used to remove morphological affixes from words, leaving only the word stem. Stemming algorithms aim to remove those affixes required for e.g. grammatical role, tense, derivational morphology leaving only the stem of the word. This is a difficult problem due to irregular words (eg. common terms in Bangla), complicated morphological rules, and part-of-speech and sense ambiguities

Model Training Approaches

Approach – 1

SignWriting: SignWriting is a system developed by Valerie Sutton in 1974, and had its roots in an early form of movement writing called DanceWriting. SingWriting is a pictorial notation system which relies on highly iconic symbols for representing hand shapes, palm orientations, facial expressions, and various indications of motion and contact. These glyphs are combined spatially on two-dimensional canvas to form individual signs as they are visually perceived.

Sutton system seems at present the best solution to the problem of SL representation as it was developed for communication purposes rather than linguistic purposes. It is now used in more than ten countries to aid literacy. Thanks to its alphabet ISWA (International SignWriting Alphabet) which includes more than 639 base symbols, SignWwriting can be used to transcribe any sign language in the world. It can record not only standard signs, but also complex constructions that are very frequent in signed discourse. This script is intuitive for new learners, but nevertheless requires mastery of a set of conventions different from those of the other transcriptions systems to become a proficient reader or writer.

SWML: SWML is an XML based format which was developed by A. Da Rocha for the storage, indexing and processing of SignWriting notation. Each sign encoded in SWML corresponds to a signbox comprising the set of symbols that together represent the notation. To identify the aspect of sign language to which it corresponds and the variation to which was subjected, each symbol is specified by a unique ID. Further, the coordinates concerning the relative position of each symbol in the bidimensional canvases are also denoted. The representation of the ASL sign “salute” in SignWriting and its SWML translation are given below in a diagram.

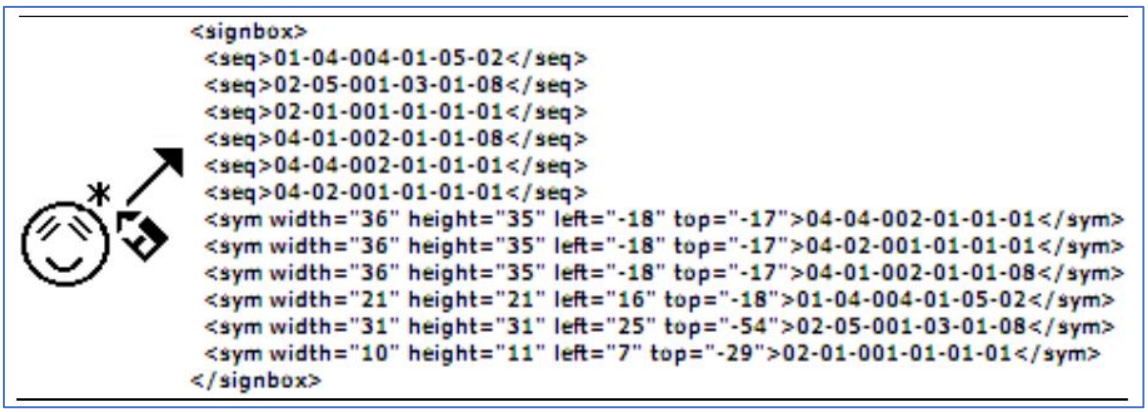


Figure 38. The SWML encoding of the sign “salute”

Indeed, there are two main drawbacks, in the SWML encoding, which make it difficult to drive automatically a signing avatar. On the one hand, there is no temporal order in which symbols are written to interpret correctly the sign. A SignSpelling Sequence can be proposed by the creator of each sign for internal ordering its symbols, but it should be notated that this sequence is an optional prefix, it cannot be found in all signboxes. For this reason, it is not possible to adopt this possible solution in our approach for arranging the notation elements. In other hand, the SWML format encodes just the original glyphs and this means that a certain amount of information may be omitted. For example, SignWriting does not use symbols for the hand location, so is SWML.

We will develop a new tool for synthesizing the gestures represented within a sign language written form in SignWriting by using a 3D signing avatar. The provision of transcribed data via a user-friendly interface, make the notation easier to read and understand by deaf and hard of hearing learners.

The input of our sign synthesis system is the SWML signbox of the SW notation to be visualized. To render its content in signed language, the set of symbols found in the signbox will be processed and converted into 3D animation sequences by implementing four steps, as shown in below diagram. The first step is dedicated to determine the correct order of symbols in the sign, while the second is devoted to provide the missing information needed to describe the movements of the avatar. The third step ensures the automatic generation of a gesture description language which explicitly specifies how sign is articulated. Finally, the last step is devoted to transform the obtained sign description to SML (Sign Modeling Language) which is interpreted then automatically by the WebSign player to animate the virtual avatar.

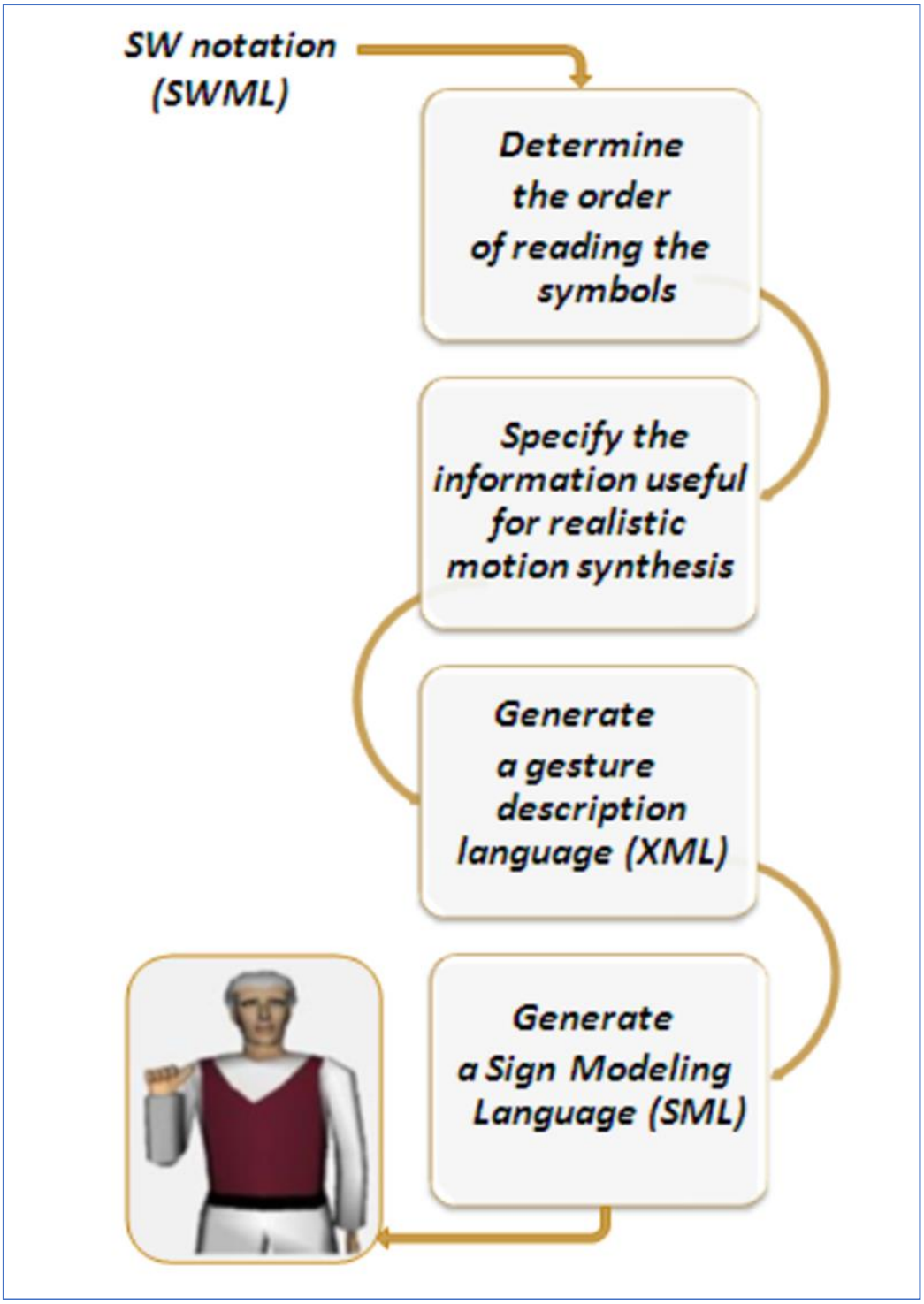


Figure 39. Proposed system process

Sorting Procedure

For the sorting process, we define the set of symbols representing hands, directional movement, finger movement and contact as an underlying structure to the sign through which we can identify its type: static or dynamic, one-handed or two-handed, symmetrical (in which both

hands have the same shape, orientation, location and movement) or not symmetrical, in unit (both hands move as a group to the same direction) or not in unit, simple or compound.

According to the type and the underlying structure of a sign, a series of rules will be applied appropriately to arrange the list symbols. For example, the sign" salute" mentioned above, is a simple sign articulated with one hand, it includes one configuration, a directional movement symbol and a touch contact. In this case, a rule based on the direction of movement arrow is used to determine if the touch contact occurs before or after the motion.

Identifying the Missing Information

Animating a 3D avatar in SL requires explicit enough information in order to be close to reality. As we have seen, SWML encoding may omit some interesting details that are implicitly present in the notation, and can affect avatar performance. The hand position is a key feature that was not explicitly defined in SW and thereby in SWML. It is rather deduced by the reader from the pictograph. To define this feature, we must always consider the most symbols' placements within the sign, especially those representing the head, shoulders, torso and limbs. But, sometimes when the hand touches different body parts (e.g. in the ASL sign "we"), it will be more suitable to use detailed location symbols. These tokens are not used to write SignWriting for everyday use, but for programming an avatar in order to give the computer information about exact locations.

Gesture Description

The purpose of the gesture description is to support the representation of signs at the phonetic level, and this by representing phonetically the significant features of signing. The gesture description we present includes essentially two segments: posture and movement. The hand posture is defined by the shape of the hand, its orientation, its degree of rotation, and also, where necessary, its position in the signing space surrounding the signer's body. In contrast, segment movement is used to specify the relevant features of any type of movements which could be a global movement, local movement or contact. The definition of non-manual components is also provided in the gesture description to specify the behavior of facial articulators such as raised eyebrows, puffed cheeks, eyes blinking, and bodily movements such as tilting the head body and shoulders.

Below diagram illustrates the description of the sign "salute" in which the tip of the right index moves to contact the upper right side of the face and then moves forward to the right in straight movement. The eyebrows of the signer are straight up and his mouth smiles. It is important to note here that the definition of all these features is based on the corresponding definition in the SignWriting alphabet (ISWA 2010).

```
<sign hands=single inUnity=false symmetry=false >
  <mouth shape=smile />
  <eyebrows shape= straight_up />
<Posture>
  <Right-hand shape=H72 orientation=FP-Side rotation=45 />
<Posture/>
<Movement>
  <Right-hand contact= touch repeat=1 part=r_index_tip_outside
              loc=r_upper_face />
<Movement/>
<Movement>
  <Right-hand globalMovement=FP_straight joint=elbow
   repeat=1 size=small direction=forward_right speed=normal />
<Movement/>
<sign/>
```

Figure 40. The description of the sign "salute"

Animating the Avatar

For rendering animations in real time, an XML based descriptive language (Sign Modeling Language or SML) is used to codify the obtained gesture description. SML was developed to provide an extra layer around X3D for facilitating the 3D virtual agent manipulation. The SML language is a successive movement of a group of joins. Every movement has a fixed time during it the rotation of every joint in the group is done. The armature design is compliant to the HI-Anim specifications, in which each joint has a specific name and specific initial orientation. Below diagram shows how left elbow joint can be rotated using SML.

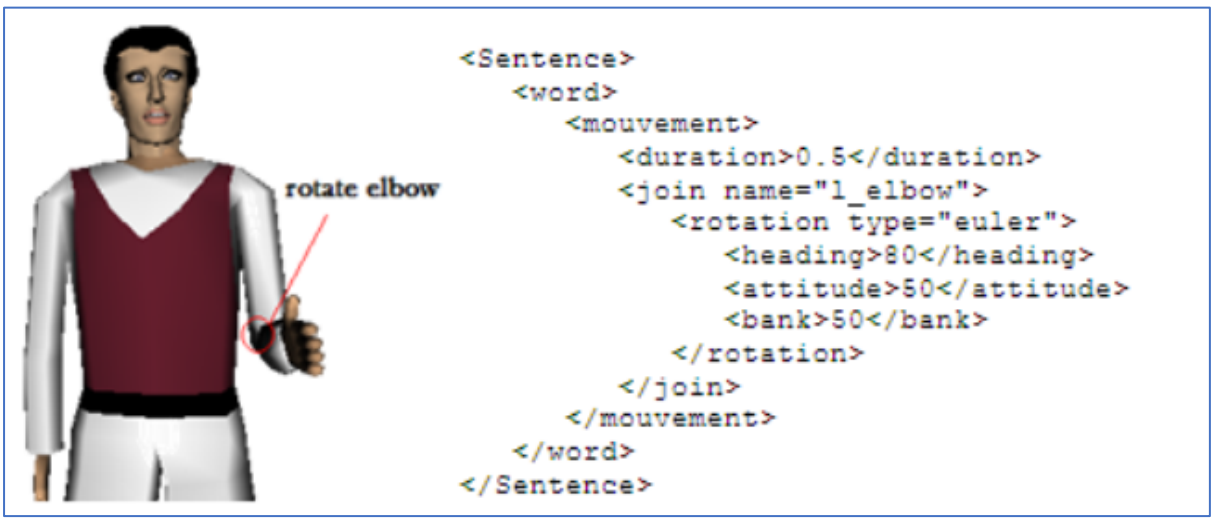


Figure 41. Rotation of elbow joint into SML

A 3D rendering module will interpret then the SML script and convert it into 3D animation using a virtual agent.

Our findings:

Sign which contains hand and mouth expression had to express in XML and also in animation mapping. So, it doesn't fit with our scenario.

Approach – 2

For the conversion and translation process of T2SP, we have constructed a stemmer model which will be known by **Spacy Model**. We have also trained 2 model which is Sentence to Gloss (S2G) and Word to Gloss (W2G).

Conversion Process

In text to sign puppet (T2SP) system, initially we will receive a normal Bangla sentence as input. Here our stemmer model which is **Spacy**, is filtering the normal Bangla sentence to a standard sentence. Now the reason of filtering the input text is, it may consist some word inconsistency and the sentence which is filtered by Spacy will remove any of incongruity.

After filtering to standard sentence from normal text input, our trained **S2G model** will receive the sentence. This model will generate the essential gloss from standard sentence. When the perfect gloss has been generated, the model connects with dataset and proceed to an approach of generative modeling which is known as **GANs**. Through **GANs**, now the Gloss can transform to sign puppet.

It's obvious that for any conditions, if our **S2G model** can't able to generate Gloss from the standard sentence, then the model will guide the sentence to word tokenizer model to handle it differently. This tokenizer will break the sentence into words and our trained **W2G model** will generate Gloss. Finally, Gloss can transform to sign puppet through the generative modeling, **GANs**.

Observations – Two of above approaches give us a clear vision of text-to-sign puppet generation method. In approach-1 the method can be found of annotation generation which helps us to build 2D and 3D animation but it's a bit difficult to work with the sign expression of hand and mouth gesture in XML. Approach-2 is more focused than approach-1 on converting the raw data to sign gloss which is another challenging part to implement but from the side of result excellency, it's better to use spaCy NLP for building several others model, which will increase the rate of accuracy for the T2SP model.

3.4.1.2 Adaptation of Sign Puppet

Rendering to web browser/Desktop agent

One of the challenges for this system is, render the animation in browser. To do that and make it compatible across the browsers we need to choose our technology wisely. So, this topic demands some sort of explanation. Currently, two technology we can consider for this and based on our progress in the work we will finalize one. They are X3D and WebGL.

X3D

In 1995, VRML became the first web based 3D format. VRML was unique because it supported 3D geometry, animation, and scripting. In 1997, VRML was ISO certified and continued to attract a large following of artists and engineers. VRML is the most widely supported 3D format for tools and viewers.
VRML has been superseded by X3D, and is a direct subset of X3D (where the X stands for Extensibility).
X3D is a XML based format for representing and communicating 3D information. It is an improved version of the VRML (or WRL) format and shares many similarities. X3D is used for 2D and 3D graphics, 3D viewers, animation, computer assisted design, navigation and much more. X3D is used on the 3D Print Exchange as an effective way to preview models before we download them. We can use an embedded viewer called X3DOM that allows models to be visualized in a web browser. We can also download X3D files of the models on the exchange from the download menu, and then import them into 3D software such as Blender or Meshlab. An advantage of X3D is that, unlike STL, it encodes color information, so downloading the X3D file is essential if you are going to print your model on a color 3D printer. We can also use downloaded X3D files for displaying 3D models elsewhere, such as targeted site, by embedding them into an X3D viewer.

Noticeable Features of X3D

- X3D is an ISO-ratified, file format and run-time architecture to represent and communicate 3D scenes and objects.
- X3D fully represents 3-dimensional data.
- X3D is developed and maintained by the Web3D Consortium.
- X3D has evolved from its beginnings as the Virtual Reality Modeling Language (VRML) to the considerably more mature and refined ISO X3D standard.
- X3D provides a system for the storage, retrieval and playback of real time 3D scenes in multiple applications, all within an open architecture to support a wide array of domains and user scenarios.

X3D has a rich set of componentized features that can be tailored for use in engineering and scientific visualization, CAD and architecture, Geospatial, Human animation, 3D printing and 3D Scanning, AR/MR/VR. Today X3D has several applications in medical visualization, training and simulation, multimedia, entertainment, education, and more.
As an open standard X3D can run on many platforms, but importantly can render 3D models in most web browsers without the requirement for additional or proprietary applications.

Further, once models are developed utilizing X3D, these easily port to alternative platforms like holographic, head-mounted or other display devices.
X3D is 3rd-generation VRML. X3D is a direct superset of VRML with four encodings: XML encoding .x3d, Classic VRML encoding .x3dv, VRML97 encoding .wrl and JavaScript Object Notation (JSON). Programming language bindings are available for JavaScript and Java, with work in progress on C++/C# and Python. X3D is designed so that all of these forms are functionally equivalent, we can choose to use any of them.

WebGL

WebGL is a JavaScript API that allows us to implement interactive 3D graphics, straight in the browser. WebGL is a web standard developed by the Khronos group. WebGL runs as a specific context for the HTML <canvas> element, which gives us access to hardware-accelerated 3D rendering in JavaScript. Because it runs in the <canvas> element, WebGL also has full integration with all DOM interfaces. The API is based on OpenGL ES 2.0, which means that it is possible to run WebGL on many different devices, such as desktop computers, mobile phones and TVs.

Using WebGL

To access WebGL content we need to have a browser that supports it.

- Opera 12.00 or above (enable webGL by entering Enable WebGL in opera:config by setting the value to 1, and Enable Hardware Acceleration similarly, then restart the browser)
- Chrome 9 or higher, on Linux, Mac and Windows
- Firefox 4 and higher
- Safari 5.1 or higher on Leopard, Snow Leopard, or Lion (make sure you enable WebGL in Safari — go to Preferences > Advanced and check “Show develop menu in menu bar”, then go to Develop > Enable WebGL)

WebGL Features

It is based on OpenGL ES 3.0 and new features include:

- 3D textures,
- Sampler objects,
- Uniform Buffer objects,
- Sync objects,
- Query objects,
- Transform Feedback objects,
- Promoted extensions that are now core to WebGL 2: Vertex Array objects, instancing, multiple render targets, fragment depth.

WebGL in action

WebGL is slightly more complicated than typical web technologies because it's designed to work directly with your graphics card. As a consequence, it's pretty low level. This is what allows it to rapidly do complex 3D rendering involving lots of calculations.
But the beauty of this thing is we don't need to fully understand the inner workings of WebGL. There are several WebGL libraries available to take some of the complexity out of our hands. However, gaining an understanding of it can be useful in case we really want to spice up our code with features that are not in our library of choice, or if we feel that having a better grasp of the technology will help us to find our way around what the libraries have to offer.
When programming in WebGL, you are usually aiming to render a scene of some kind. This usually includes multiple subsequent draw jobs or "calls", each of which is carried out in the GPU through a process called the rendering pipeline.
In WebGL, like in most real-time 3D graphics, the triangle is the basic element with which models are drawn. Therefore, the process of drawing in WebGL involves using JavaScript to generate the information that specifies where and how these triangles will be created, and how they will look (colour, shades, textures, etc). This information is then fed to the GPU, which processes it, and returns a view of the scene.

The rendering pipelines

Note that this section is adapted from Joe Groff's explanation of the graphics pipeline in OpenGL.

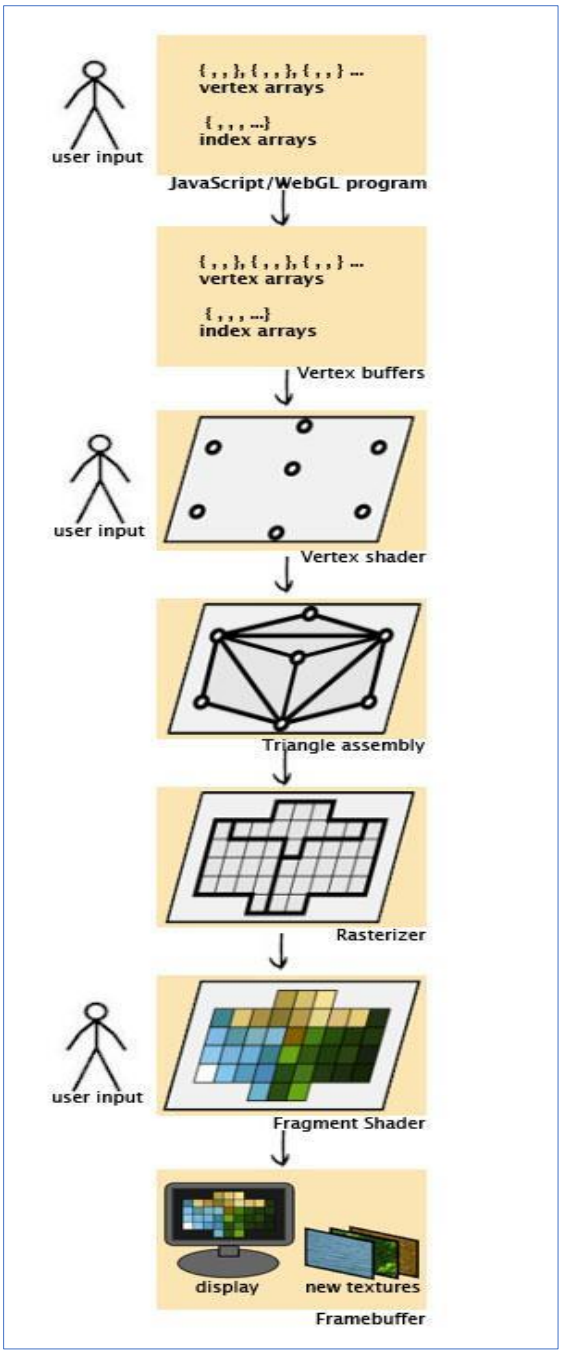


Figure 42. Animation rendering workflow using WebGL

The process starts with the creation of the vertex arrays. These are arrays that contain vertex attributes like the location of the vertex in the 3D space and information about the vertex' texture, color or how it will be affected by lighting (vertex normal). These arrays and the information they contain are created in JavaScript in one or more of these ways: processing files that describe a 3D model (for example .obj files), procedurally creating the data from scratch, or using a library that provides vertex arrays for geometrical shapes.

Then the data in the vertex arrays is sent to the GPU by feeding it into a set of one or more vertex buffers. When a rendering job is submitted, we also have to supply an additional array of indices that point to the vertex array elements. They control how the vertices get assembled into triangles later on.

Choosing a WebGL library

Some of the notable libraries we can use for are:

- Three.js (Github) is a lightweight 3D engine with a very low level of complexity — in a good way. The engine can render using <canvas>, <svg> and WebGL. This is some info on how to get started, which has a nice description of the elements in a scene. And here is the Three.js API documentation. Three.js is also the most popular WebGL library in terms of number of users
- PhiloGL (Github) is built with a focus on JavaScript good practices and idioms. Its modules cover a number of categories from program and shader management to XHR, JSONP, effects, web workers and much more. There is an extensive set of PhiloGL lessons that can go through to get started. And the PhiloGL documentation is pretty thorough too.
- GLGE (Github) has some more complex features, like skeletal animation and animated materials. You can find a list of GLGE features on their project website.
- J3D (Github) allows not only to create our own scenes but also to export scenes from Unity to WebGL. The J3D "Hello cube" tutorial can help you get started. Also have a look at this tutorial on how to export from Unity to J3D.

As mentioned, we can also write our own WebGL from scratch, using no libraries.

3.4.1.3. System Architecture (T2SP)

T2SP is another core module for our proposed framework. Distinctive stages of the module will perform distinctive exercises. They can be isolated into taking after wide bunches. Detail discourse on each area is portrayed at afterward area in a consecutive way.

Our proposed system architecture is based on sign modeling language data base. The system takes the input text as a request. It will perform some initial text processing like POS tagging, filtering, stemming and create output to be an input to the Text Adapted Bangla Sign Modeling Language module. After this operation we will obtain a text form adapted to the sign language specificities and which represent an input to WebSign kernel module. Finally, this engine will generates the sign modeling language SML according to the given text form to be interpreted by our virtual character.

- Text Processing
- Prepare text adaptive Bangla Sign Language
- Develop dataset for rendering puppet animation
- Render animated puppet

- Testing and result evaluation

Overall system architecture of the T2SP conversation is depicted into the below figure:

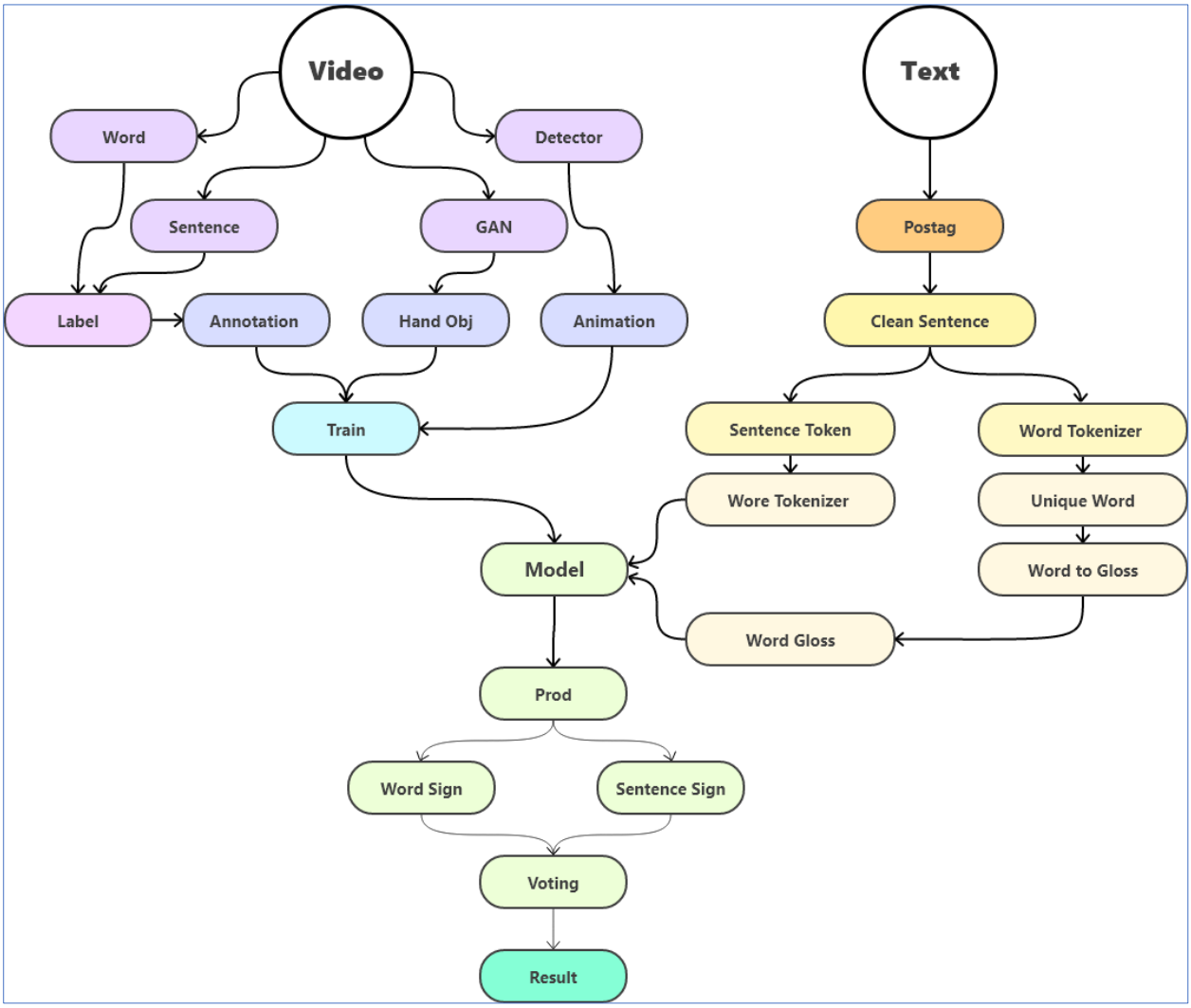


Figure 43. High Level system architecture of T2SP

Render animated puppet

The flow chart of the virtual agent program development is depicted below. The development process includes modeling, texture mapping and VR programming. The rendering would be done using X3D and WebGL. The output of the previous stage would be passed this stage to render the corresponding animation in end users application like web, mobile, or desktop.

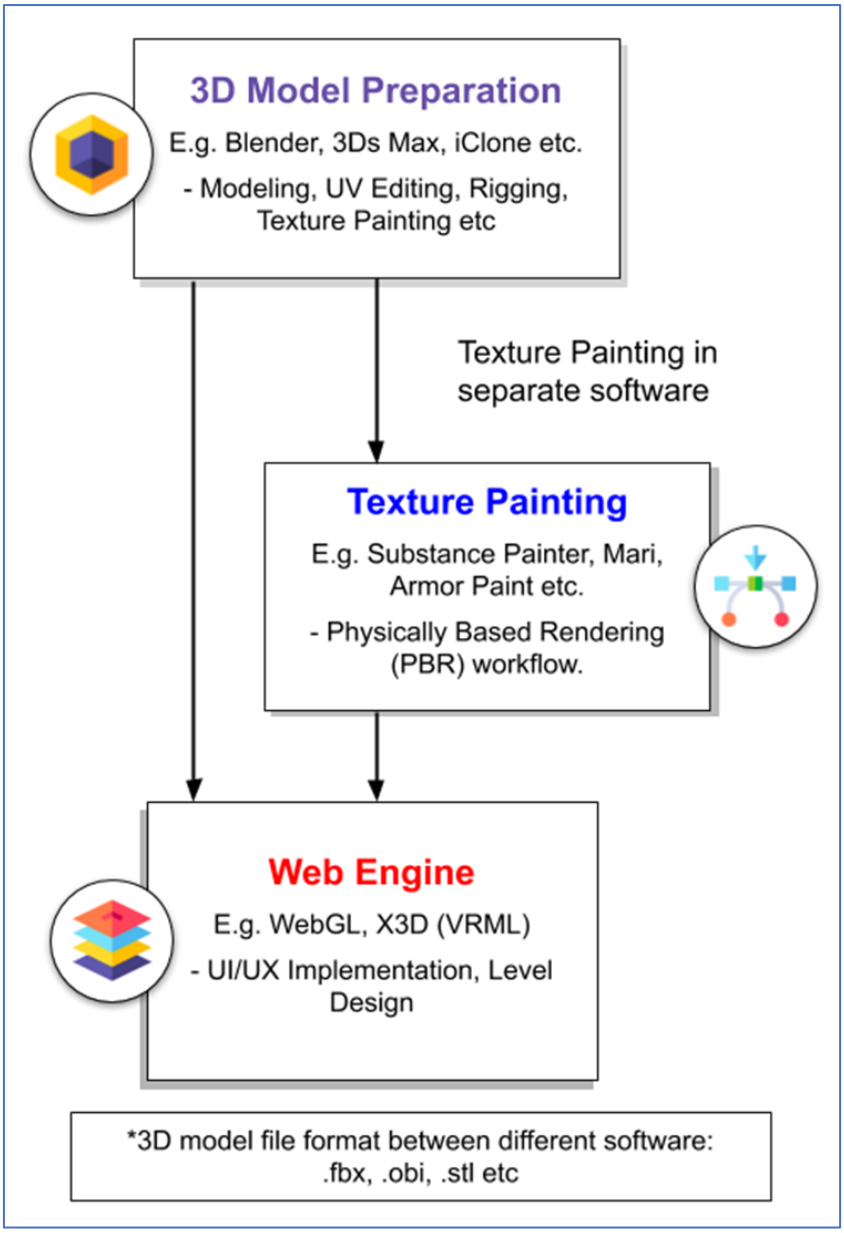


Figure 44. Workflow of preparing 3-D model

3.4.1.4. Testing & Evaluation

T2SP GANs puppet & multimodal

Validation curves

As already discussed in the previous section, the goal with any machine learning model is generalization. Validation curves allow us to find the sweet spot between under-fitting and over-fitting a model to build a model that generalizes well.

A typical validation curve is a plot of the model's error as a function of some model hyper-parameter which controls the model's tendency to over-fit or under-fit the data. The chosen parameter depends on the specific model you're evaluating; for example, you might choose to plot the degree of polynomial features (typically, this means you have polynomial features up to this degree) for a linear regression model. Generally, the chosen parameter will have some degree of control over the model's complexity. On this curve, both the training error and the validation error have been plotted for the model. Using both of these errors combined, a diagnosis can be carried out whether a model is suffering from high bias or high variance.

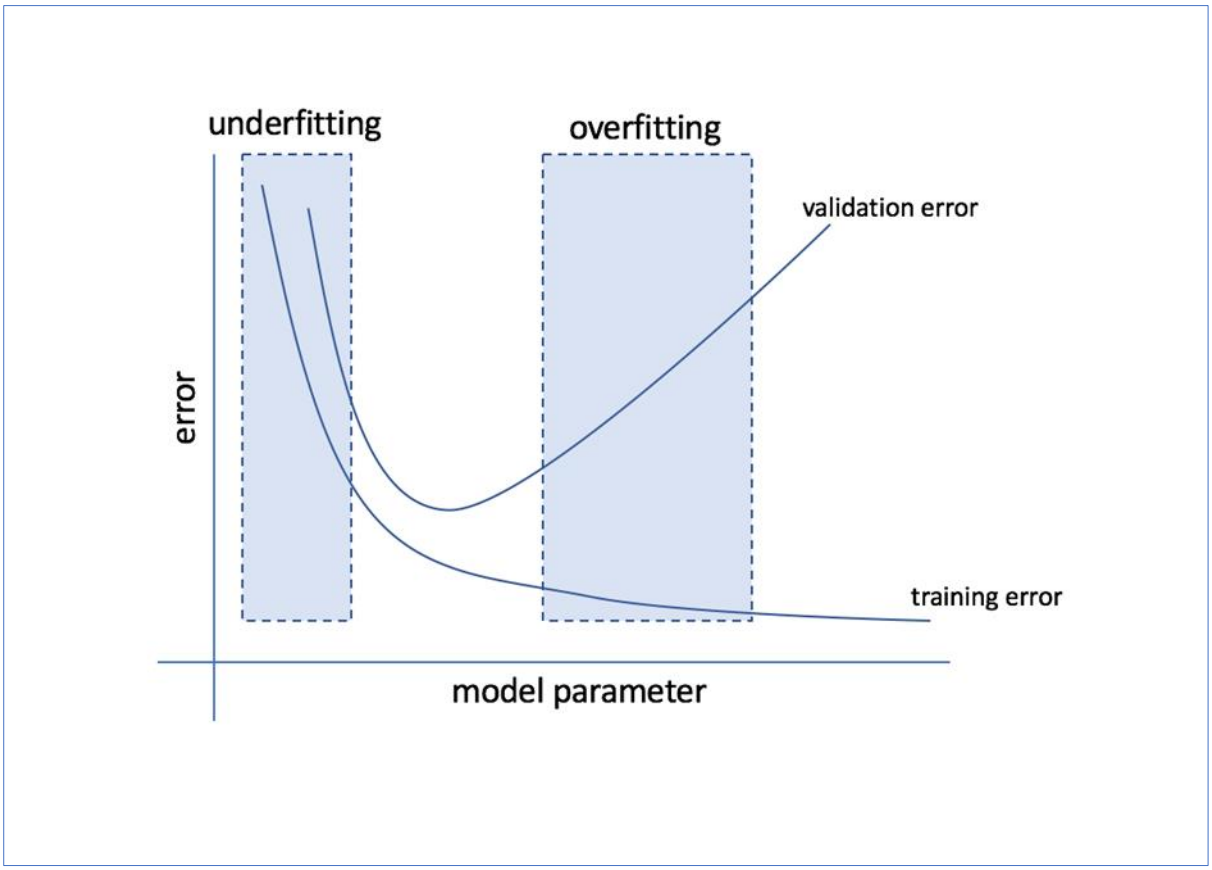
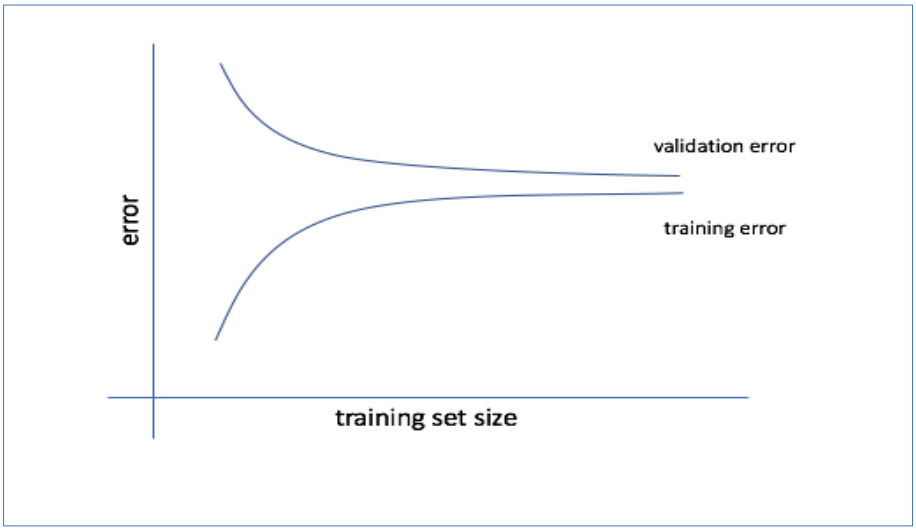


Figure 45. Validation Curve

Learning curves

The second tool comes in discussion for diagnosing bias and variance in a model is learning curves. Here, we'll plot the error of a model as a function of the number of training examples. Similar to validation curves, we'll plot the error for both the training data and validation data.

If our model has high bias, we'll observe fairly quick convergence to a high error for the validation and training datasets. If the model suffers from high bias, training on more data will do very little to improve the model. This is because models which underfit the data pay little attention to the data, so feeding in more data will be useless. A better approach to improving models which suffer from high bias is to consider adding additional features to the dataset so that the model can be more equipped to learn the proper relationships.



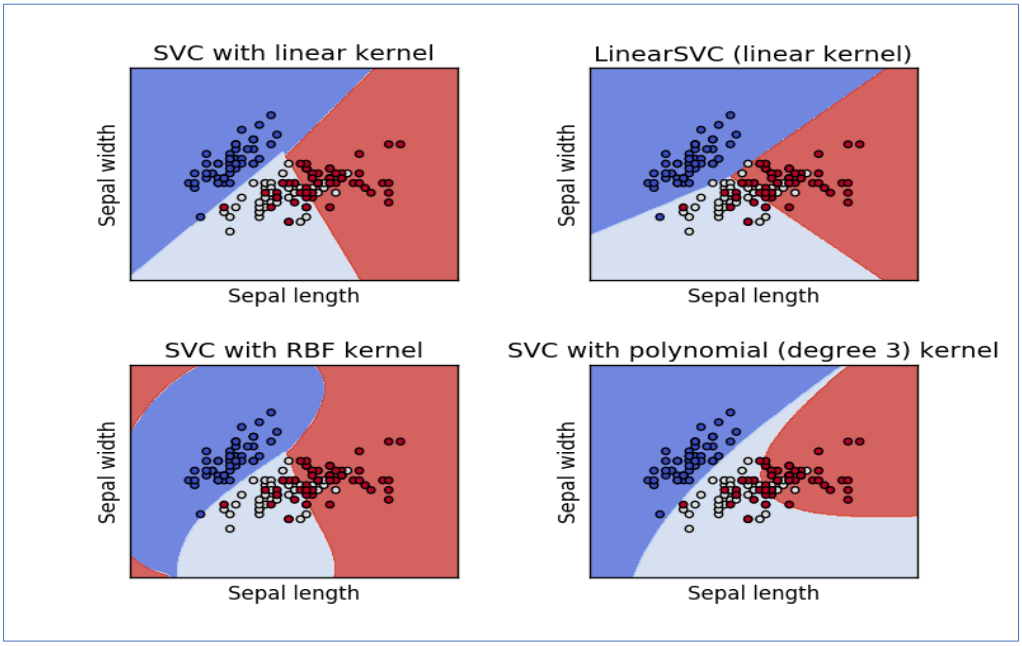


Figure 46. Learning Curve

Tracking Procedure

The captured static markers on the head (while sentence level video was captured), the facial markers used to generate an individual 3D model of a participant's face, and the emotion markers will be compared with the generated output (text to avatar). This procedure ensures that the movement of the markers is measured with high precision and independently of head movements. In addition, raw motion data that was captured as well as scaling data are exported for further standardization and statistical analyses.

- **Tracking avatar head movements** - The static head markers have to be tracked to obtain information about head movements. In addition, the several different facial surface markers must be tracked for the short parallax phase of the video clip. Subsequently, from the parallax displacement of the tracked markers for the different frames of the video clip, 3D coordinates for each tracked marker need to be measured.

- **Tracking the Avatar facial surface** - The individual 3D surface of the participant's (while recording video) face is built on the basis of the 3D coordinates of the several different surface markers. First, the facial surface is constructed by connecting four markers at a time to form rectangles - the connection lines between the facial surface markers. Subsequently, this rough approximation of the facial surface by the rectangles is interpolated and smoothed to closely fit the participant's real facial surface. Afterward, the tip of the nose in the facial surface is centered at the origin of Blenders 3D coordinate system. This way all the recorded data path is compared with the generated avatar to find the close approximation.

- **Tracking Avatar emotion markers** - The facial emotion markers must be tracked for the emotionally relevant episodes from the video clips. The movement of these markers is also exported into respective tool like Blender's 3D space, namely as a projection from the moving camera onto the static facial surface. The movement of these markers on the facial surface represents the movement of the markers painted on the participants 'skin. A python script uses Blender's application programming interface (API) to access the 3D coordinates of the emotion markers per frame, exports the coordinates, and saves them in a comma-separated values file (CSV) for each participant for further data processing. A comparison can be made with the generated avatar and the recorded clip when 3D model was made.

- **Scaling and standardization procedures** - Two individual scaling procedures might be followed for the generated modeling data to be rescaled into meaningful measurement units. The process of taking the data from the tracks of the video clip and transforming them into Blender's 3D space (A modeling tool) cannot be controlled and is therefore relatively arbitrary. Therefore, accurate measurement of a real-world object of known size is needed for the rescaling. Different techniques will be followed for rescaling.

1. **Accuracy of the measurement procedure** - To estimate the accuracy of the tracking procedure, need to calculate the following parameters:

- Accuracy of measuring head movements and estimating the facial surface.
- Accuracy of measurement of the scaling parameters.
- Accuracy of the model building and rescaling

3.5 Approach for Bangla to Braille Converter (B2BC)

The core functionality of a B2BC is to convert Bangla Unicode text input in various file formats to Bangla braille. As the undertakings that are normal as result from this task are profoundly specialized, and in most cases, are important for dynamic examination, thorough investigation of the significant writing is needed for acquiring the normal result out of this undertaking. We have perused a lot of condition of the craftsmanship writing that arrangement with executions of those errands, wrote by the eminent scientists, and have chosen a portion of those to follow for execution. In spite of the fact that further R&D will be required, and immediately done, during the execution of the undertaking, this early investigation try will give us a decent beginning stage towards finishing of the difficult specialized assignments required.

3.5.1 Desktop Application

3.5.1.1 Braille converter with editor

For Bangla Unicode text to Bangla Braille converter (B2BC), the following approach has been proposed:

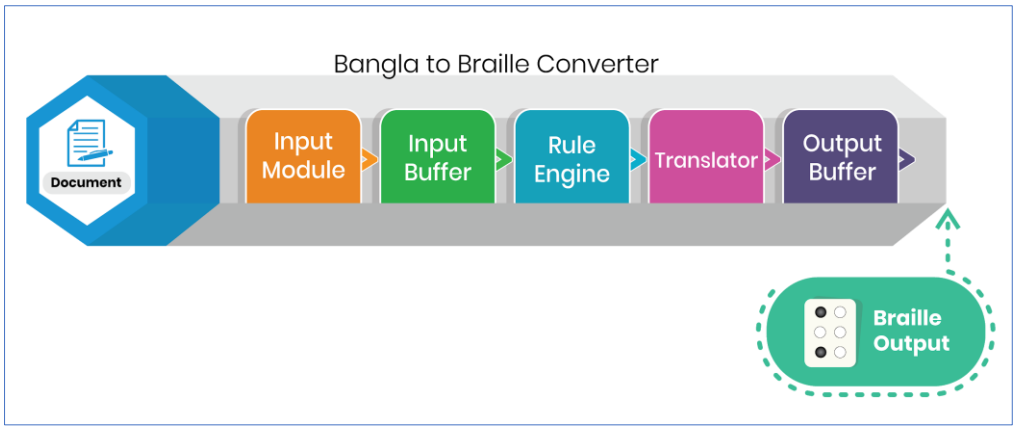


Figure 47. Proposed Model

Grammatical Conversion and Mapping Rules

In conversion of Bangla text to the corresponding Braille code, this is necessary to deal with the conjunctions, consonants, dependent and independent vowels, punctuation and numbers. The different grammatical conversion rules are discussed with examples below.

Replacing Rule: In replacing rule, each Bangla character is replaced by its corresponding Braille cell(s) (Pramait, 2001). Some uses of consonants, vowels (dependent and independent) and punctuations are given in below diagram.

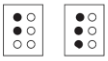
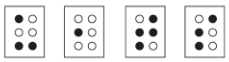
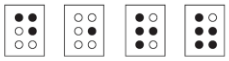
Bangla Word	Distribution	Braille Representation
বল	ব ল	
উৎস	উ ৎ স	
কাঠ	ক া ঠ	
দৃঢ়	দ ৃ ঢ়	
কমা	,	
সেমি কোলন	;	

Figure 48. Use of consonants, vowels and punctuation

Inserting Rule: In inserting rule, a Braille cell is inserted as prefix. Other characters are replaced according to replace rule (Pramait, 2001). If there is " ই ", " উ ", or " ও " after consonant and if " অ " is pronounced, then Bangla " অ " equivalent Braille cell dot 1 is inserted after consonant in Braille. Examples are shown in below Figure.

Bangla Word	Distribution	Braille Representation
বই	ব ই	
রওনা	র ও না	

Figure 49. Insertion of 'অ'

Braille cell dot 4 is inserted before conjunctions having combination of two letters (Pramait, 2001). Examples are shown in below Figure.

Bangla Word	Conjunct	Distribution	Braille Representation
গ্রাম	গ্র	গ ্ র	
পূর্ব	র্ব	র ্ ব	

Figure 50. Conjunction of 2 letters

Braille cells dot 4, 6 are inserted before conjunctions having combination of three letters or four letters (Pramait, 2001). Examples are shown in below Figure.

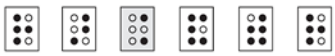
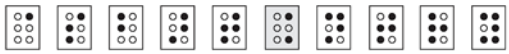
Bangla Word	Conjunct	Distribution	Braille Representation
রাষ্ট্র	ষ্ট্র	ষ ্ ট ্ র	
স্বাতন্ত্র্য	ন্ত্র্য	ন ্ ত ্ র ্ য	

Figure 51. Conjunction of 3 or 4 letters

Two Bangla conjunctions have direct representation in Bangla Braille. So there is no need to use Braille cell dot 4 before them (Pramait, 2001). They are shown in below Figure.

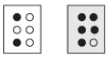
Bangla Word	Conjunct	Distribution	Braille Representation
কক্ষ	ক্ষ	ক ্ ষ	
জ্ঞান	জ্ঞ	জ ্ ঞ	

Figure 52. Two conjunctions without prefix

Braille cell dots 3,4,5,6 are inserted before the first digit of a number (Pramait, 2001). Examples are shown in below Figure

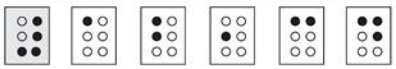
Bangla Number	Braille Representation
১২৩৪	
১২,৩৪	

Figure 53. Number Rule

Sample of B2BC Desktop Application & Editor

In below there is some pictures of Bangla to Braille converter application with editor:

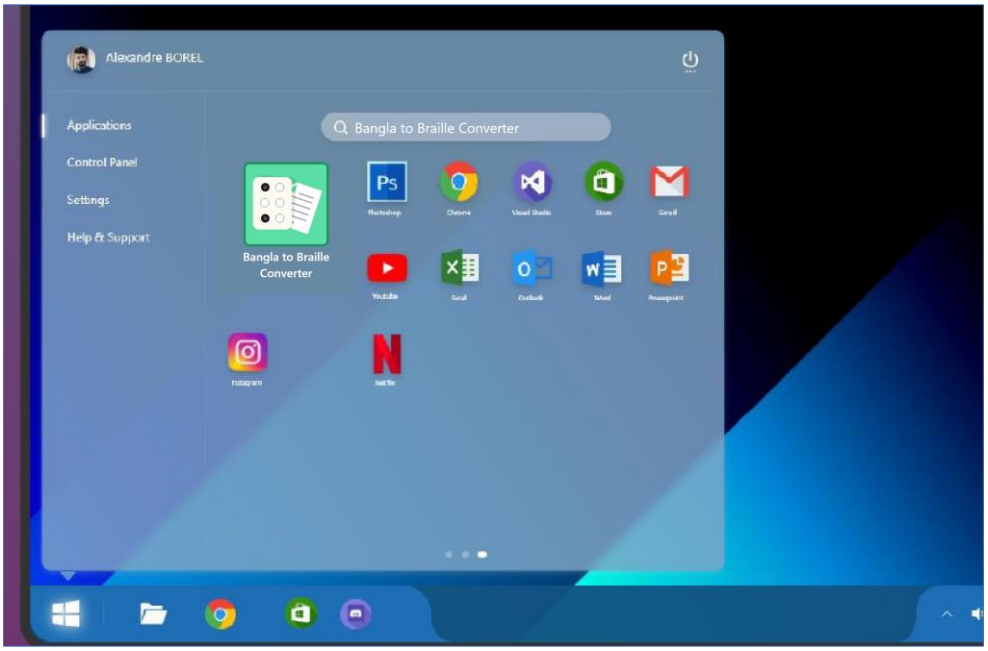


Figure 54. B2BC Desktop Application

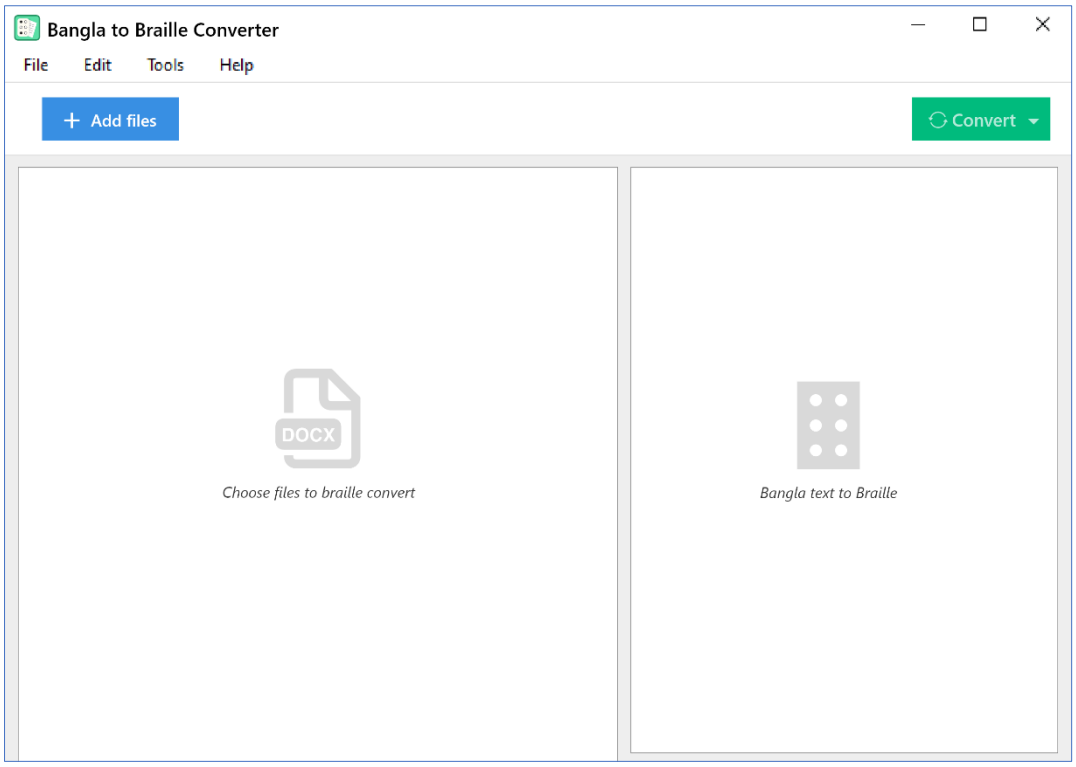


Figure 55. B2BC Converter Panel

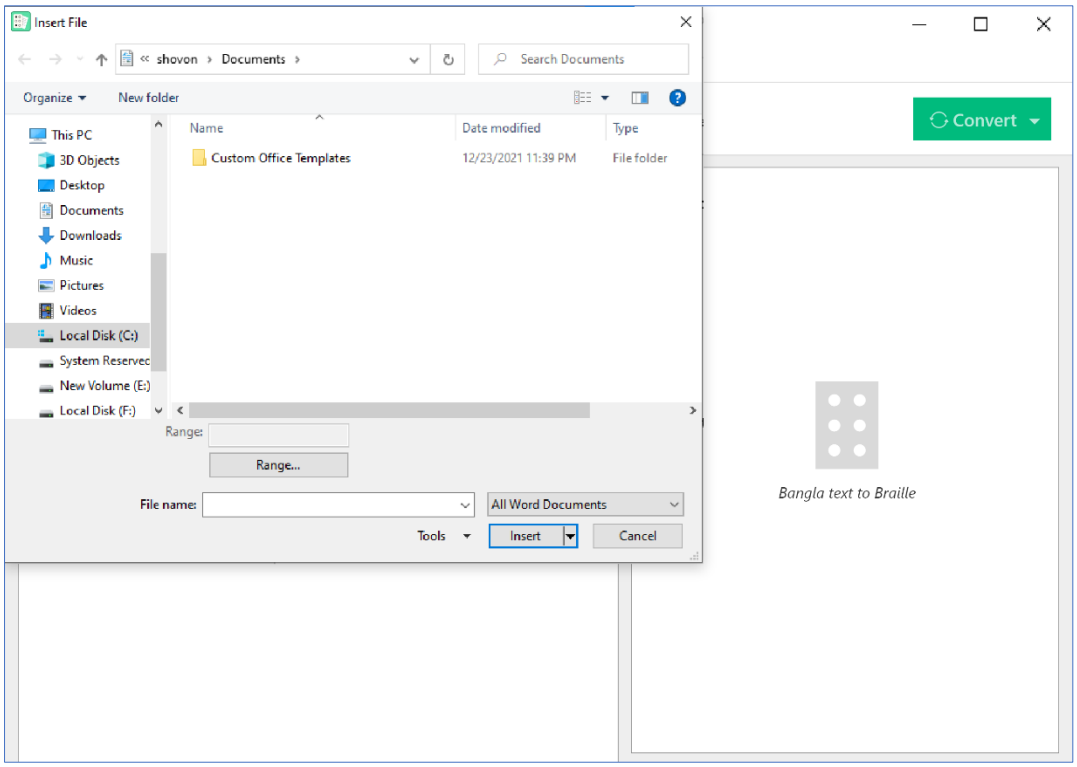


Figure 56. Uploading Text File in B2BC converter

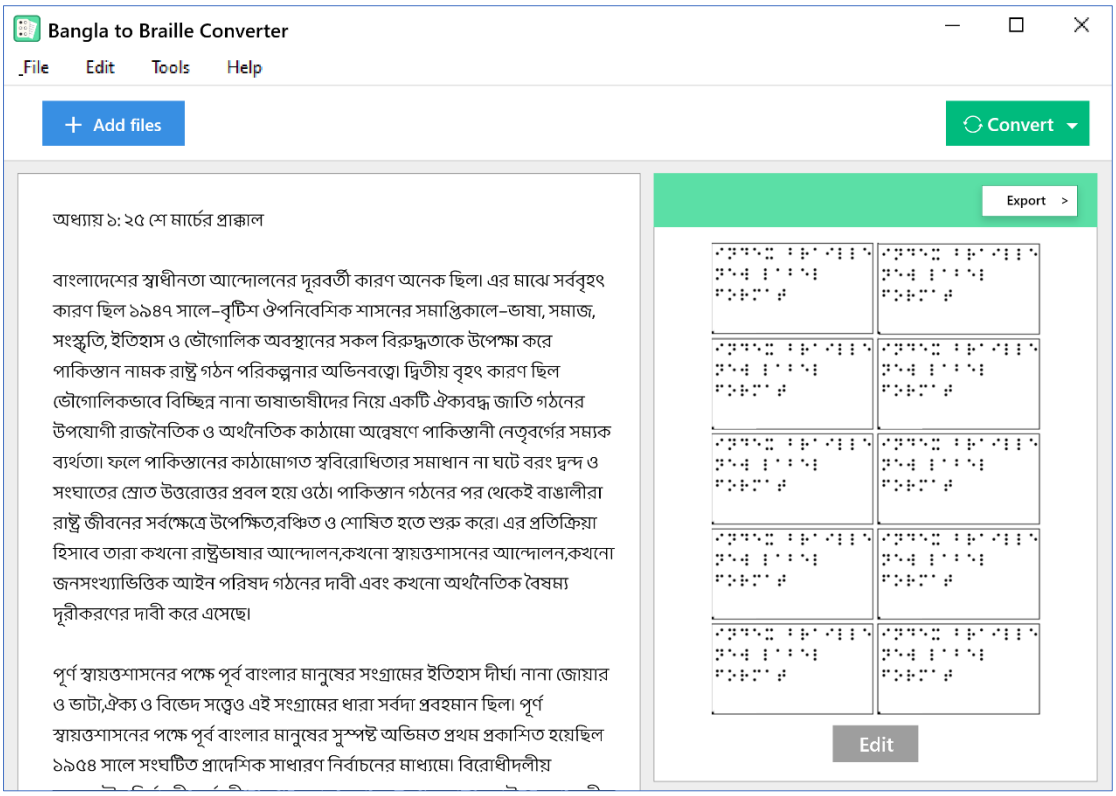


Figure 57. Bangla to Braille Converter desktop application with editor

1.3.5.1.2. System Architecture

Presentation layer

The presentation layer contains components needed to interact with the user of the application. Examples of such components are web pages, rich-client forms, user interaction process components, etc.

Service layer

Exposes the business functionality of the application allowing better support of multiple client types.

Business layer

The business layer encapsulates the core business functionality of the application. Simple business functions can be implemented using stateless components whereas complex long running transactions can be implemented using stateful workflows. The business components are generally front ended by a service interface that acts as a facade to hide the complexity of the business logic.

Data Access Layer

The data access layer provides a simple API for accessing and manipulating data. The components in this layer abstract the semantics of the underlying data access technology, thus

allowing the business layer to focus on business logic. Each component typically provides methods to perform Create, Read, Update, and Delete (CRUD) operations for a specific business entity. Business layer will fetch Mapping and Rule information's from this module

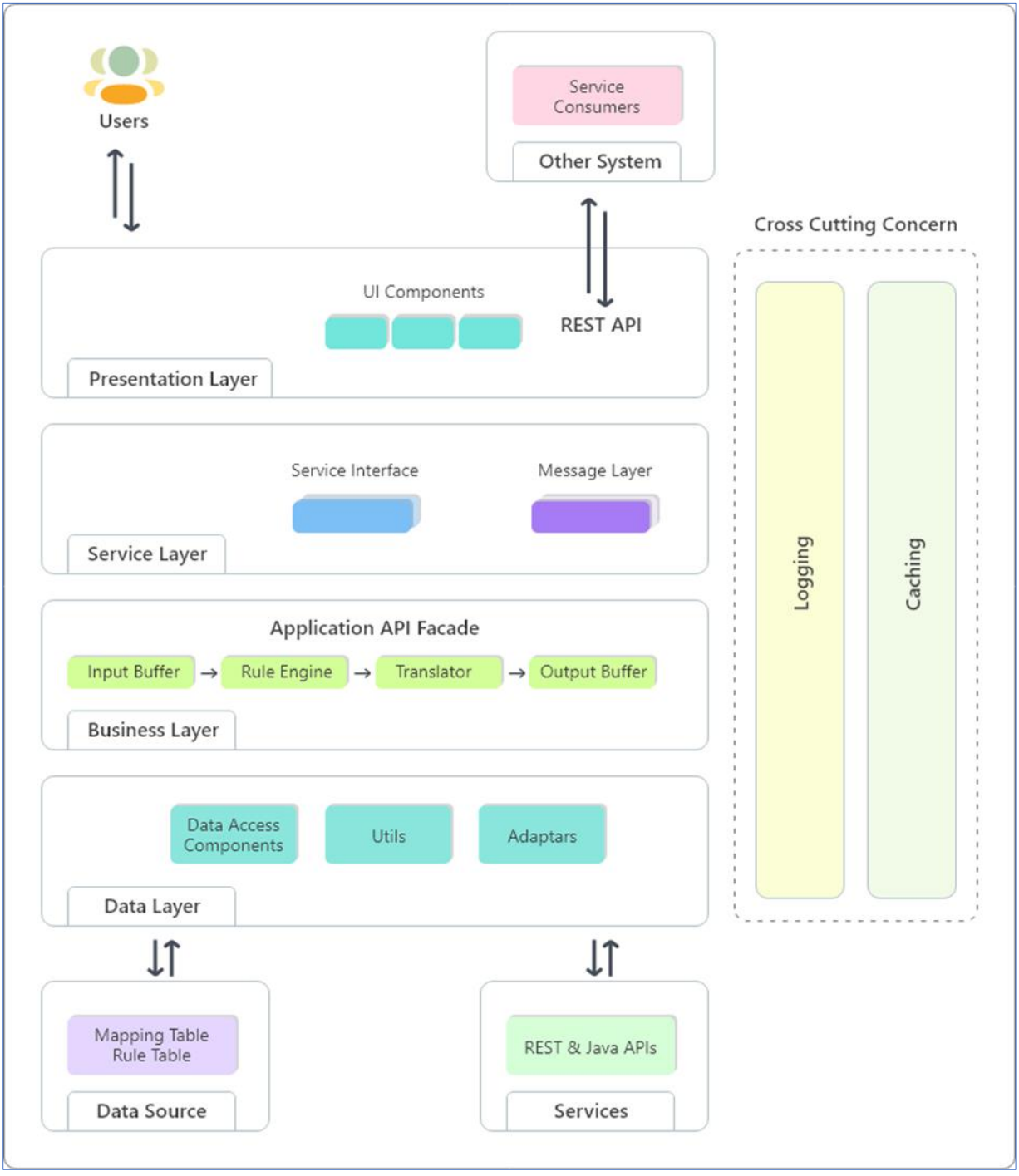


Figure 58. Application Architecture

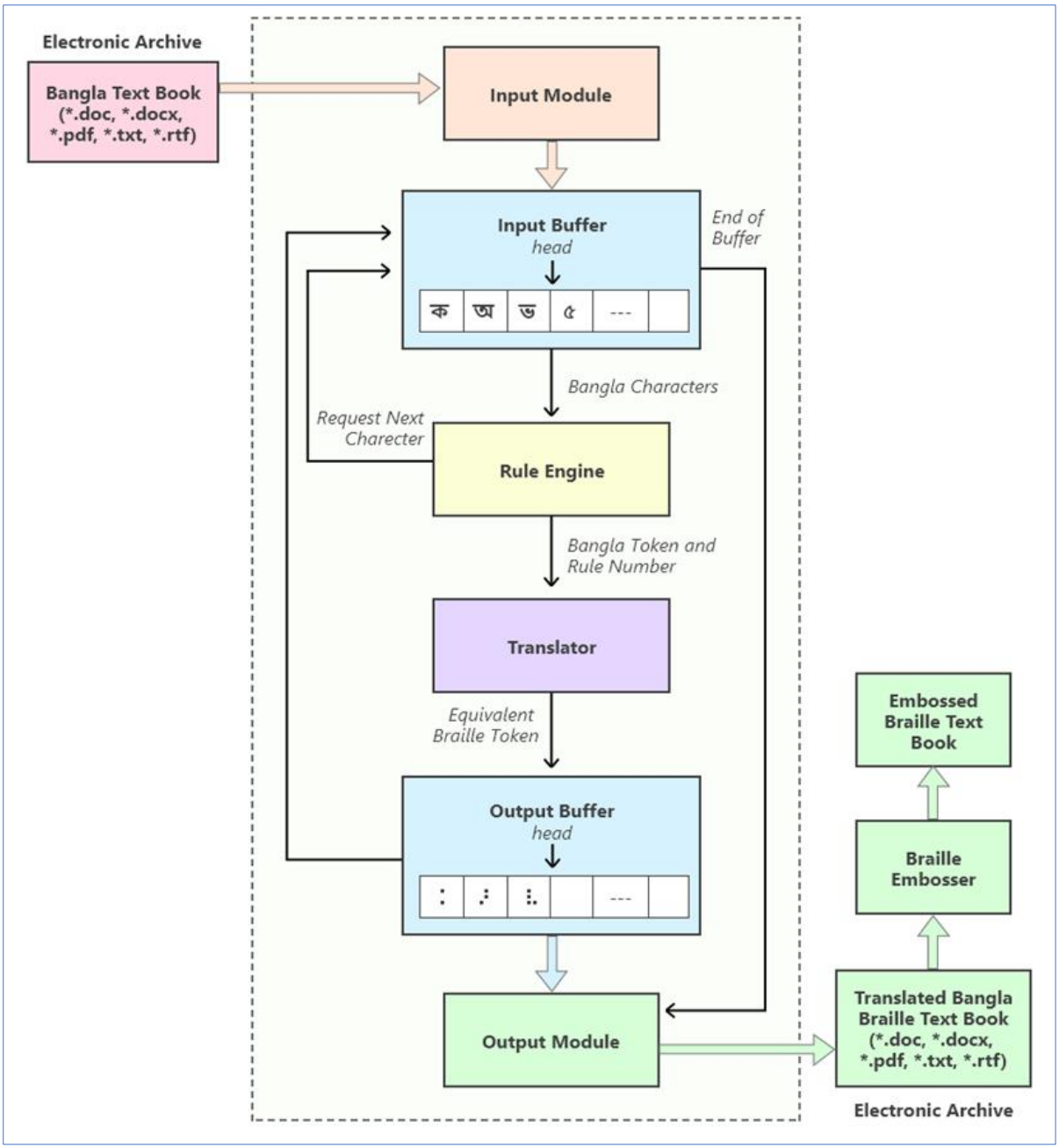


Figure 59. Core Mechanism

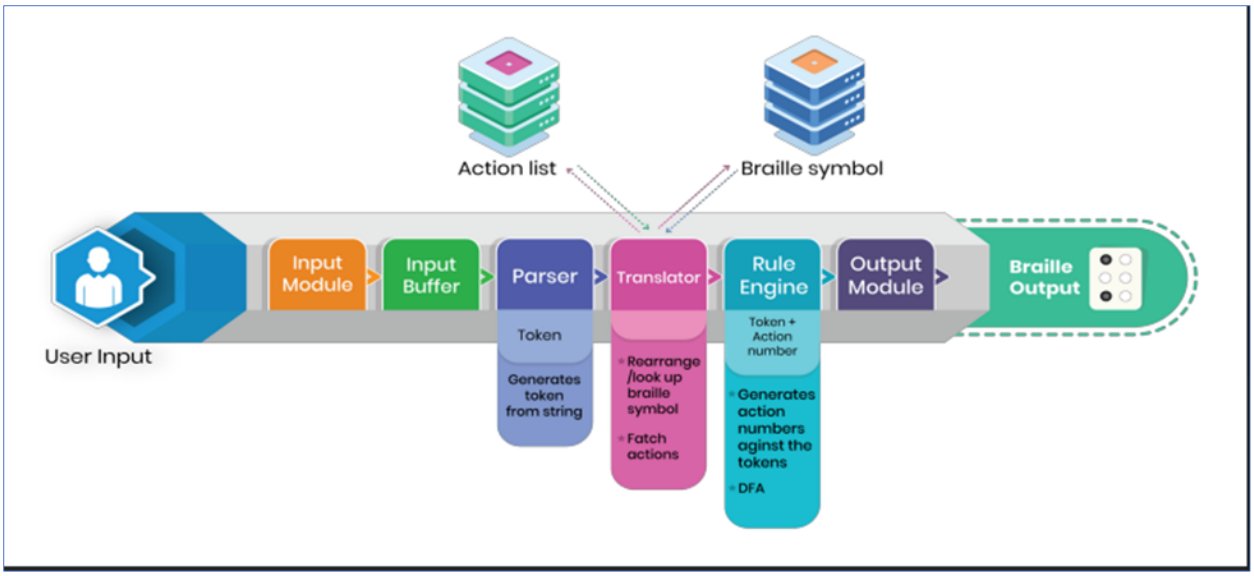


Figure 60. Core Mechanism simplified

Input Module

The main objective of input module is to read electronic Bangla text from the archive. The archive may contain in different formats like doc Microsoft Office Word (doc, docx), Open Office (doc), Text Documents (txt) and Rich Text Format (rtf). This module is responsible for detects the format of the file and acts accordingly. It will parse the Bangla text from the file. This input Unicode text will go in various steps. The core responsibility of this module is to parse the Bangla text from different file formats and pass it to others module to Braille conversion.

Input Buffer

Another core component of this converter is Input buffer. The Input buffer module is the data structure where data and related information are stored. It's more likely to linked list data structures. It contains a head, which primarily points to the initial position of the buffer. Based on the request it sends character from current head position and points the head to next position. When it reaches at end of the buffer it generates a signal indicating there is no more data to be processed.

Rule Engine

The rule engine is the central building block of this system. Rule engine generates tokens of characters are parsed and applied the rules based on the character of Unicode Bangla text. The following is a translation of the Bengali text into Braille rules (Pramait, 2001). A string of characters satisfies one of the translation rules as called token. This module reads the character of Input buffering module and by translation rule, it determines whether a sequence of character

leads to a token or not. If it forms a token, then the token and the rules it follows are sent to the next module. If not, it will send a request to the input padding module for the next character and review.

Translator

This module basically looks up on Bangla character to braille character mapping table and as well rule table where all grammatical rules have been placed. The core responsibility is to translate Bangla character to Braille character with the appropriate grammatical rules.

Output Buffer

This module is similar like input buffer module. It maintains similar data structure like Input Buffer. It's mainly receiving the translated Braille tokens from the translator module then stores in memory. It also contains a head, which primarily points the initial position of the buffer and changes its position when it gets Braille tokens from Translator. After getting token form Translator head points the next position in the buffer where the last token was placed.

Output Module

The core responsibility of this module is to collect all translated Braille Symbols from Output buffer and generate the translated Braille output into Braille text document based on user's choice. It continuously receives the translated Braille symbols from Output Buffer. It's terminating of receiving Translated Braille symbols from output buffer when it gets an indication of the end of buffer from Input Buffer. User can choose file formats like Portable Document Format (pdf), Microsoft Office Word (doc, docx), Open Office (doc), Text Documents (txt), and Rich Text Format (rtf). User can print the translated braille text with renowned Braille Printer. Output module will generate necessary signals for braille Printer.

Braille Translation Process

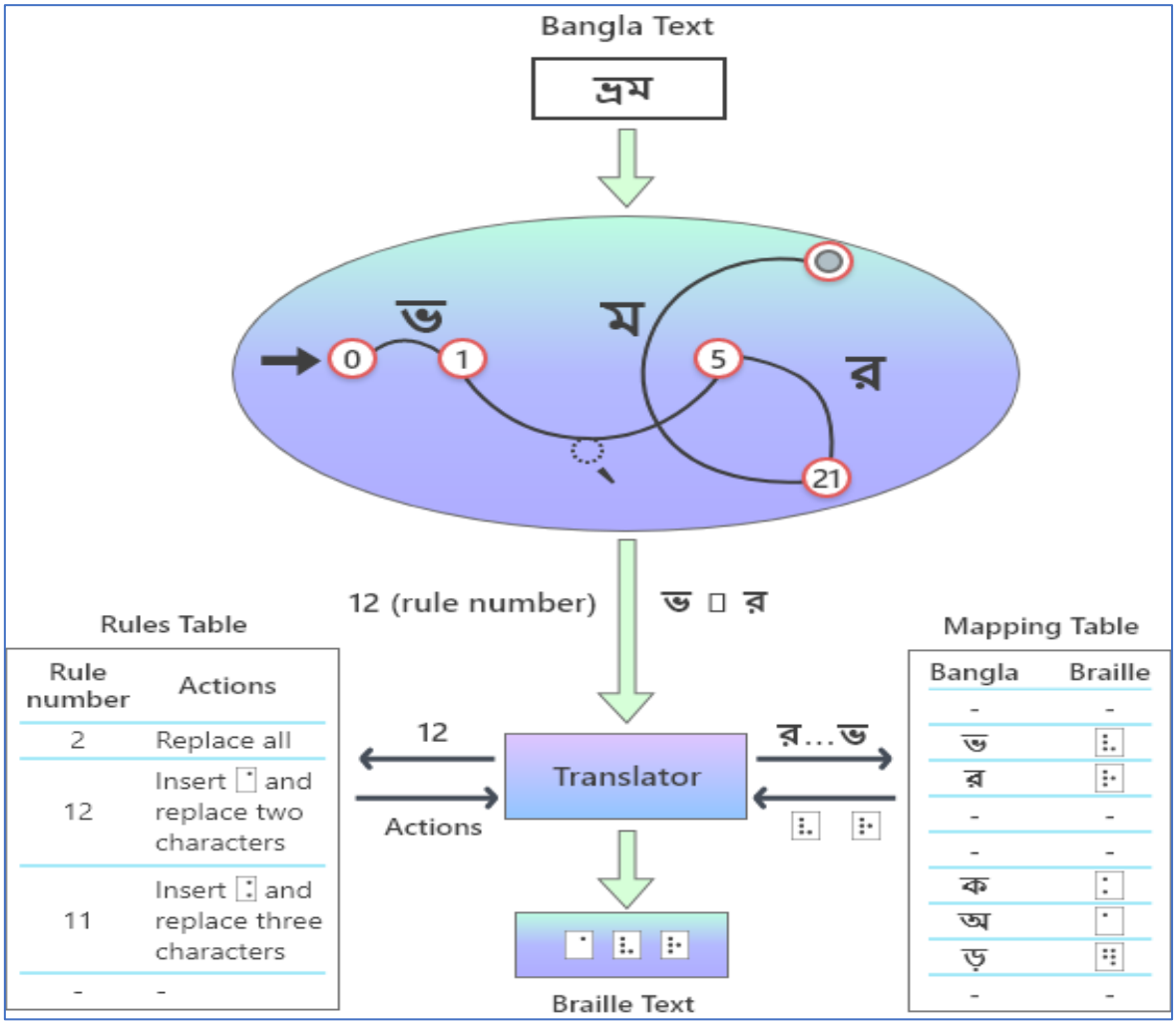


Figure 61. Braille Translation Process

The proposed computer model implementation architecture shows how the model translates Bengali text into Braille. Above is an illustrative example of how to translate a Bangla word ভ্রম into its corresponding Braille representation. At first, the word is passed to the rule engine. The rule engine is actually a deterministic finite state machine that generates a token (ভ) and a number of accept states (12) representing a specific number of rules. After the states are defined, the token and rule number are passed to the compiler. Using the rule number, the translator looks up the corresponding action for the rule number. Here, the action is to insert point 4 (⠈) and replace two characters (ভ, র). The translator then inserts point 4 and looks for the braille symbols "ভ" and "র" in the respective table to replace them. Finally, the braille symbols for are generated. Likewise, by repeated use of Bangla text ভ্রম is translated into Braille representation.

Accepting States	Description
2	Single Bangla character is replaced by equivalent Braille cell.
4	A consonant is replaced and then Braille cell dot 1 is inserted.
10	All Bangla characters are replaced by equivalent Braille cells.
11	Braille cell dot 4, 6 is inserted and three Bangla characters are replaced by equivalent Braille cells.
12	Braille cell dot 4 is inserted and then two Bangla characters are replaced by equivalent Braille cells.
13	Single character ending with "**hashanta**" are replaced.
14	Braille cell dot 4 is inserted and then two Bangla characters ending with "**hashanta"** are replaced by equivalent Braille cells.
15	Braille cell dot 4, 6 is inserted and three Bangla characters ending with "**hashanta"** are replaced by equivalent Braille cells.
19	Numbers are replaced in equivalent Braille cells with number prefix inserted at first.
22	Braille cell dot 4 is inserted and then two Bangla characters are replaced by equivalent Braille cells and Braille cell 1 is inserted at last.
24	Braille cell dot 4, 6 is inserted and then three Bangla characters are replaced by equivalent Braille cells and Braille cell 1 is inserted at last.
25	Braille cell dot 4, 6 is inserted and then four Bangla characters are replaced by equivalent Braille cells.

Table 12. Accepting States of Braille Translator

1.3.5.1.3 API

Sample Endpoint (API) of B2SP converter

The system will expose some endpoint for other services to integrate with braille converter. One of them is Office ADD-IN plugin.

API	Endpoint	Description
Translate Bangla text to Bangla Braille	https://b2sp-prod.com.bd/text/bn/convert	
Translate English text to English Braille	https://b2sp-prod.com.bd/text/en/convert	
Translate Bangla File to Bangla Braille	https://b2sp-prod.com.bd/file /bn/convert	

Table 13. Sample API of Braille

Note: The above Endpoints have no real existence as of now, they are for example purpose only.

3.5.2. Office Extension

Office Add-in

Office Add-ins platform used to build solutions that extend Office applications and interact with content in Office documents. With Office Add-ins, you can use familiar web technologies such as HTML, CSS, and JavaScript to build a solution that can run in Office across multiple platforms including Windows, Mac, iPad, and in a web browser. In Office Add-ins platform

we have option to build Add-in to extend default features for all the components of Office suite like Word, Excel, PowerPoint, Outlook, Visio etc. In this project we need to develop a MS Word extension of B2BC converter.

Figure 62.High Level Overview of Microsoft Office Add-In

With Word add-ins, we can use familiar web technologies such as HTML, CSS, and JavaScript to build a solution that extends the functionality of Word and can run in Word across multiple platforms including Windows, Mac, iPad, and in a web browser. Word add-ins are one of the many development options that you have on the Office Add-ins platform. We will use add-in commands to enable Braille printing features in MS Word. A very High-level overview of Microsoft Office (Word).

Figure 63. Overview of MS Word Add-In of B2BC.

Best practices for developing Office Add-ins

Effective add-ins offer unique and compelling functionality that extends Office applications in a visually appealing way. We will apply the best practices described in this section to create add-ins that help users to print Bangla Unicode text document to Braille printing output quickly and efficiently.

- We will create add-ins that help user's complete tasks quickly and efficiently. Focus on scenarios that make sense for Office applications.
- The Add in will be embed complementary services within Office hosts.
- It will improve the Office experience to enhance productivity.
- We will make sure that the value of your add-in is clear to users right away by creating an engaging first run experience.
- Create an effective AppSource listing. Make the benefits of your add-in clear in your title and description. Don't rely on your brand to communicate what your add-in does.
- Engage new users with a highly usable and intuitive first experience. Note that users are still deciding whether to use or abandon an add-in after they download it from the store.
- We will make the steps that the user needs to take to engage with your add-in clear. Use videos, placemats, paging panels, or other resources to entice users.
- Will reinforce the value proposition of your add-in on launch, rather than just asking users to sign in.
- We will provide teaching UI to guide users and make your UI personal.

Sample of Office Add-ins for B2SP

An office add-in will be developed which will appear in Word's Ribbon Bar like below figure. User needs to press on Braille option from the dropdown which eventually generated corresponding translated file. In the background this add-in places some REST call to our exposed API from B2SP core module. The documented text will be posted with REST body as multipart data. Then the text will go various phases of our core module's input buffer, rule engine, translator and the output module. Then the multipart translated data can be downloaded in user's define locations. That's how the Add-ins will works. It may take sometimes based of the size of the input text.

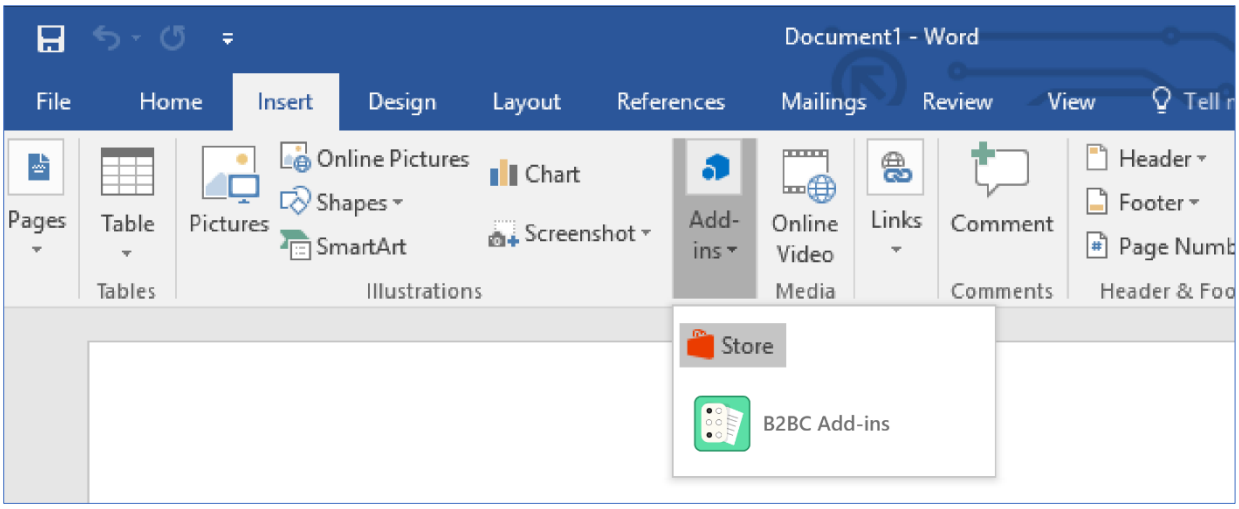


Figure 64. B2BC Office Extension

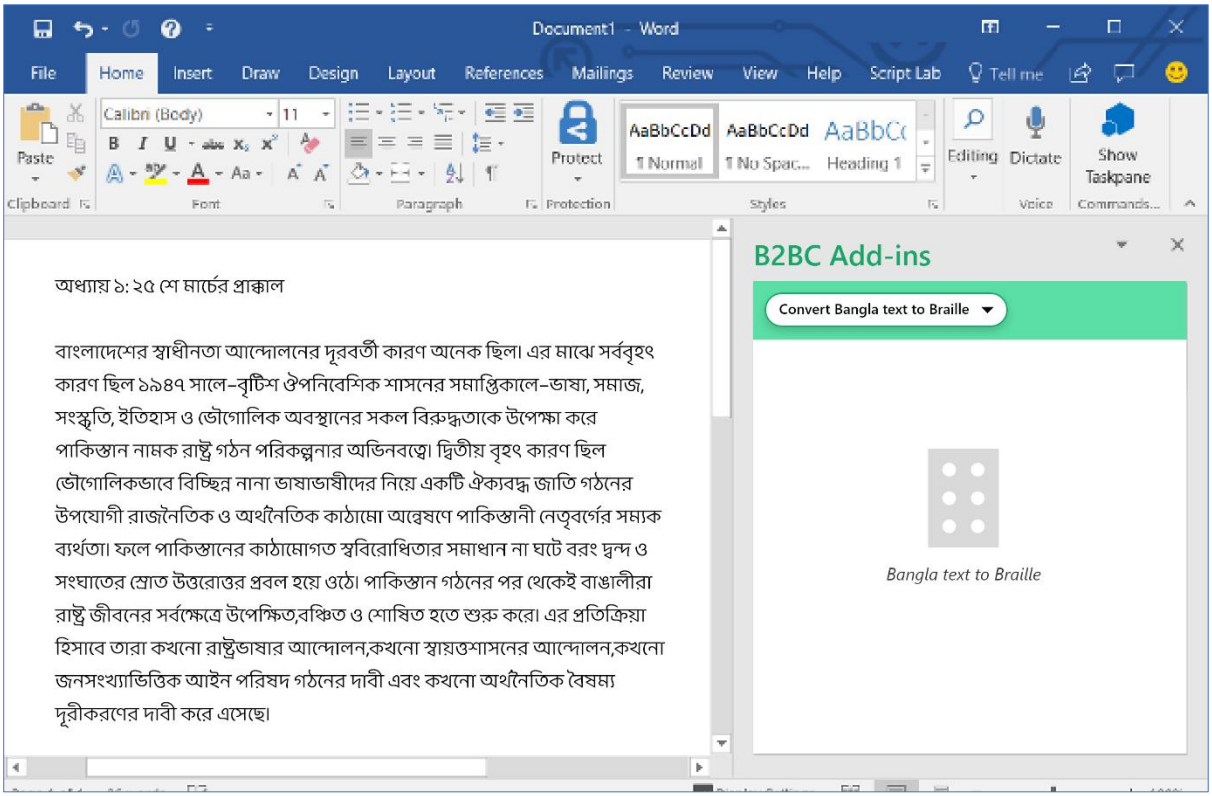


Figure 65. Uses of Braille Extension before Conversion

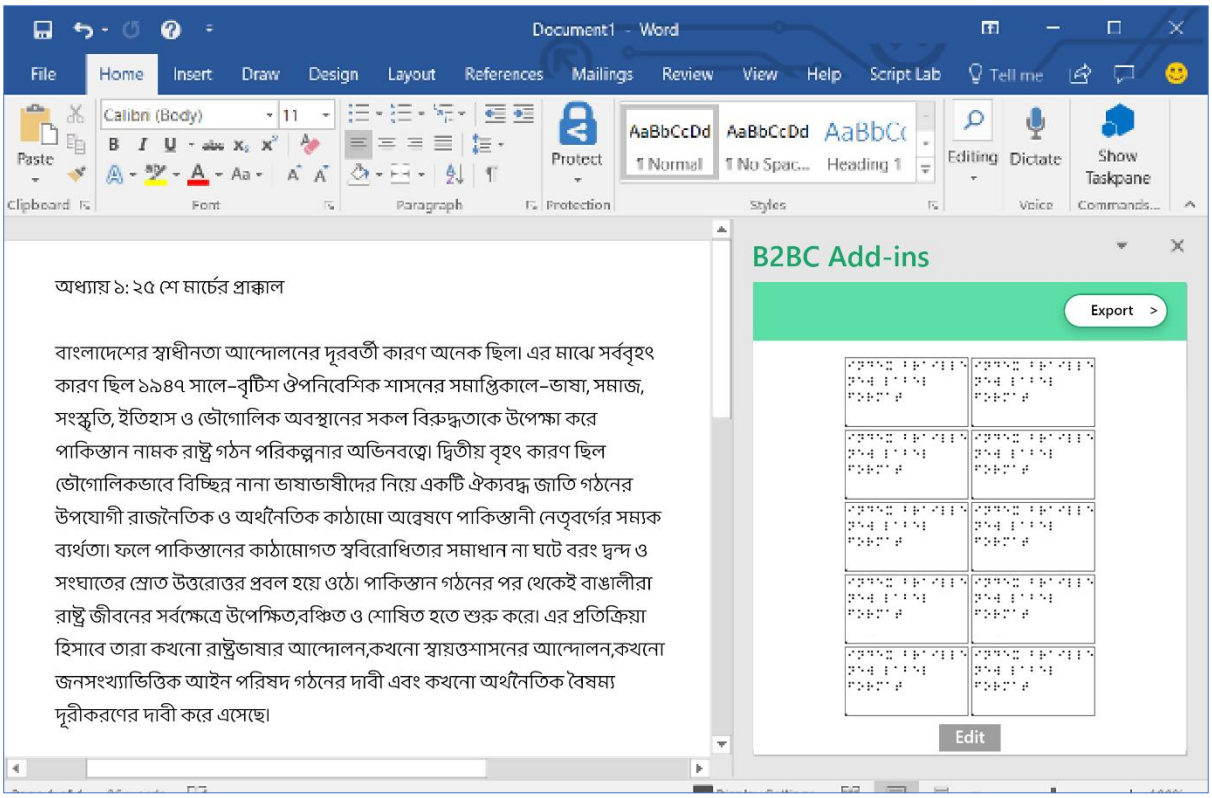


Figure 66. Uses of Braille Extension after Conversion

3.6 Approach for Mobile Application

3.6.1 Approach & Methodology of SLR Mobile Application

Good architecture is first and foremost a profitable architecture that makes the process of developing and maintaining a program simpler and more efficient. A program with a good architecture is easier to expand and change, and also to test, debug, and understand. For example, the Clean Architecture is a good fit for large scale projects with big budgets. This type of architecture is universal, allowing for installation of various plug-ins and quick troubleshooting, but it should not be created using frameworks. Program code must be written from scratch.

Let’s look at our proposed implemented mobile application architecture:

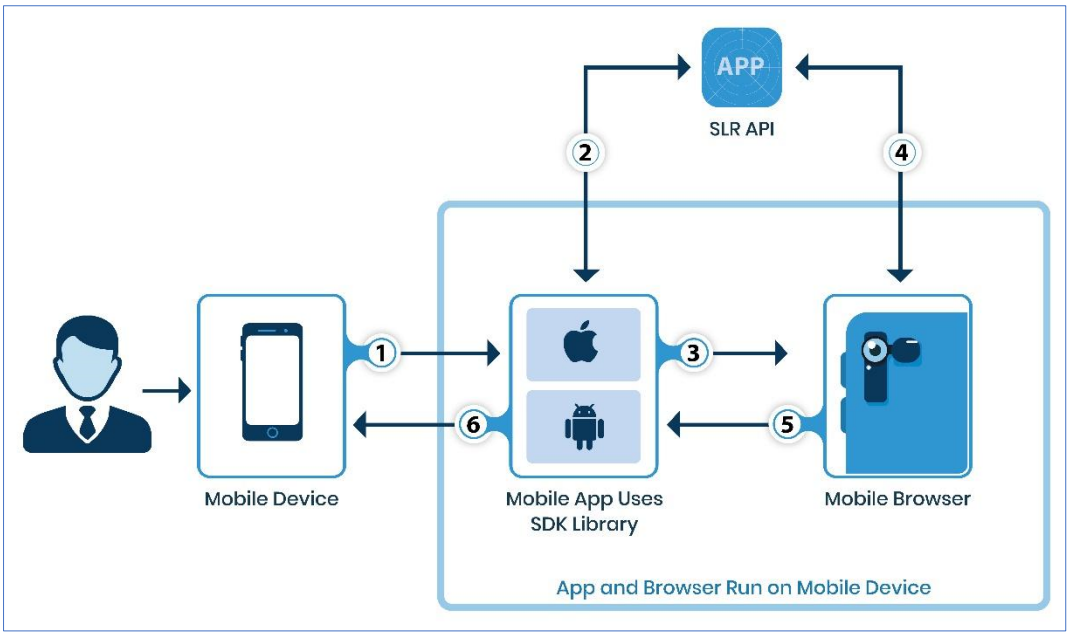


Figure 67. Mobile Application Architecture (high-level)

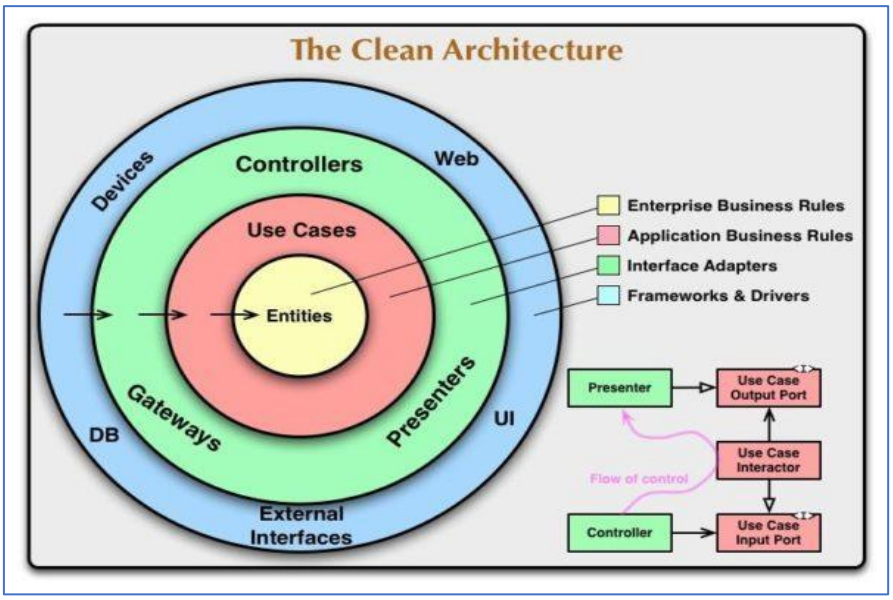


Figure 68. Mobile Application Architecture (low-level)

Each layer of such a mobile application is independent of other programs and components and entitles is a key fragment containing the logic of our proposed application and important objects. All layers are connected by the Dependency Rule, which states that in the source code all dependencies can only be specified internally. For example, nothing from the outer circle can be mentioned by a code from the inner circle. This applies to functions, classes, variables, or any other entity.

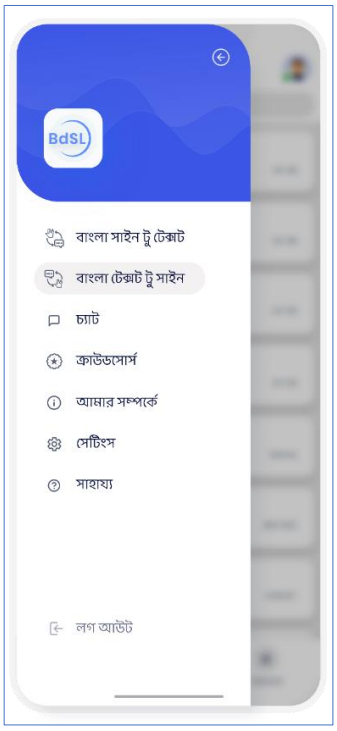
3.6.2 Mobile User Interface for BdSLR Application

Based on the design first approach - the possible mockups screen for mobile apps for LSR has been designed. The following subsequent screen shots list some windows for mobile apps – including:

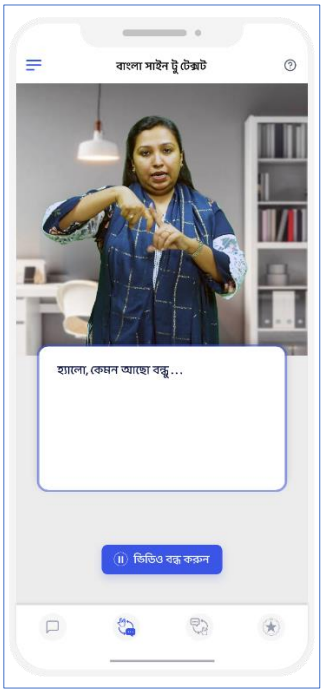
1. Splash Screen
2. Screen with Left Menu
3. Contacts Window
4. Option to select chat type (Text to sign or Sign to Text)
5. Chat window for text to Sign
6. Chat window for Sign to Text
7. Bangla Sign to Text chat window (deaf person is performing sign to someone)
8. Window for Puppet/Avatar (someone has sent text to deaf)

Splash page: Splash page is the first opening impression of Bangla Sign Language (BdSL) apps in mobile.

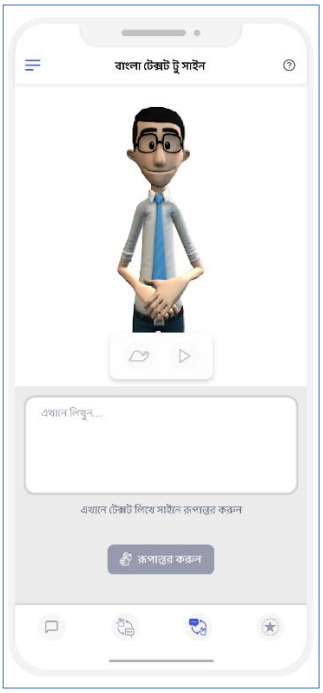
Left Menu: If we click the BdSL icon after opening, then a menu panel will slip over screen from left side. This menu will show all the available feature from this apps.



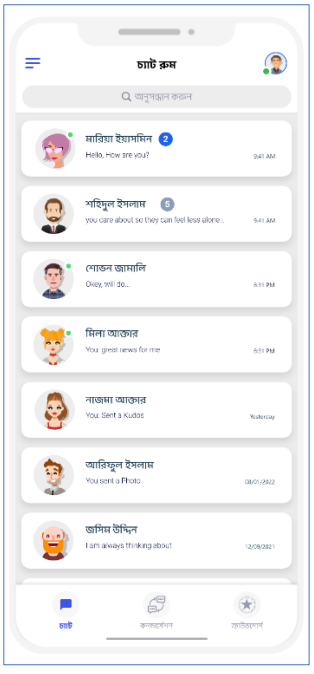
Bangla Sign to Text: In this feature screen, normal text can be seen in the text field if a user shows the sign language through the rare camera or front camera.



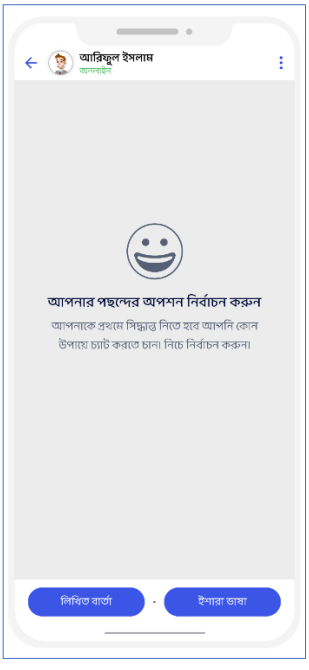
Bangla Text to Sign: This is a feature screen where an animated puppet can be seen acting the sign language for any specific sentence written by a user in text field.



Chat Contact: All the contact can be seen in this panel with whom, the logged user has been chat previously.

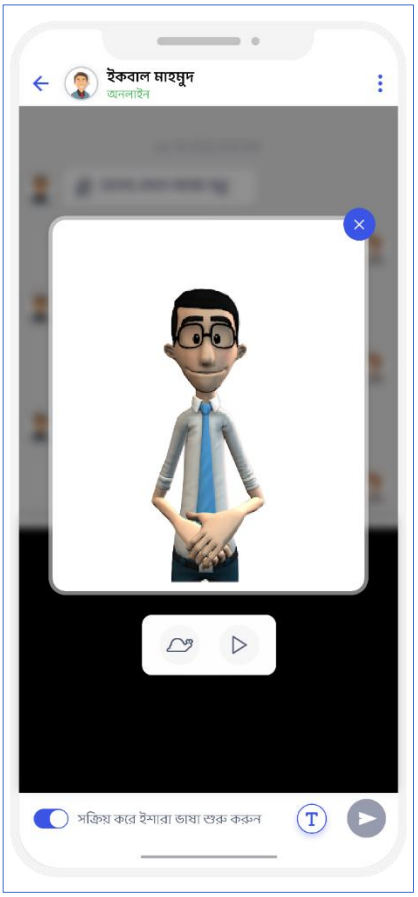


Chat-Set Preference Way to Chat: When a registered user wants to chat, this feature will ask to choose any of an option that the user prefer for. "ইশারা ভাষা" & "লিখিত ভাষা are the two choices from where the user can take either one.



Sign Mode Chat from Deaf User Screen: In this screen, a both side parallel communication can be seen in between a registered hearing disable user and a regular user. The sign mode chat screen shows on the side of hearing disable user where chat communication is in a threaded view and the Bangla text sent by regular user converts(T2SP) and shows as a sign puppet animation on the below panel. A deaf user can act sign language through either front or rare camera. The sign language will converts (SLR) to Bangla text and seen by regular user.

Sign Play to Touch Text Bubble from Deaf User Screen: In sign mode chat screen, a hand icon bubble can be seen besides the text of regular user. If a deaf user wants to check the text anytime in future, he/she can touch that text bubble which will show the sign puppet animation for the corresponding text.



Text Mode Chat from Regular User Screen: The text mode chat screen shows on the side of regular user where Bangla text is written. The text convert into sign puppet and displays as animation on the side of hearing disable user.

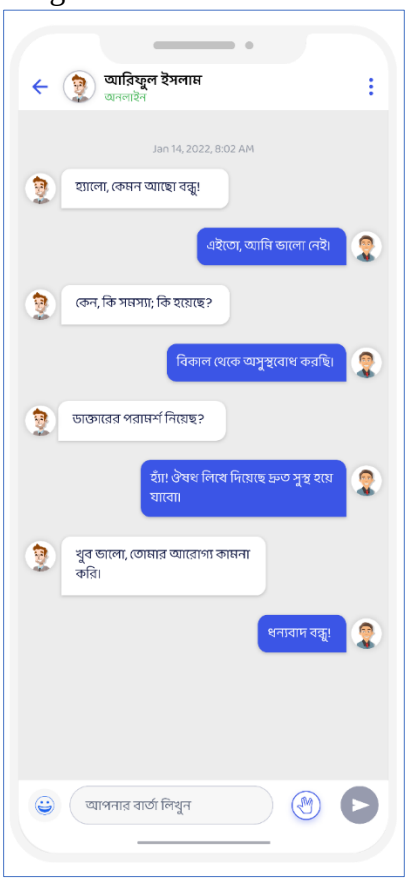


Figure 69. Samples of BdSL Mobile application screens

3.7 BdSL enabled communicator (Chat)

3.7.1 Approach & Methodology of BdSL enabled web-based Communicator for chat

Using web browser or mobile application deaf and dumb user can register/login to the chat application. Proposed system will access webcam/ Mobile device camera to record the sign language video while messaging. User may perform sign language for a full sentence or a single gloss and using SLR API, proposed chat application will recognize the sign language and translate to corresponding Bangla Unicode text. User can see the translated text is input screen and will be able to send that message to other recipients.

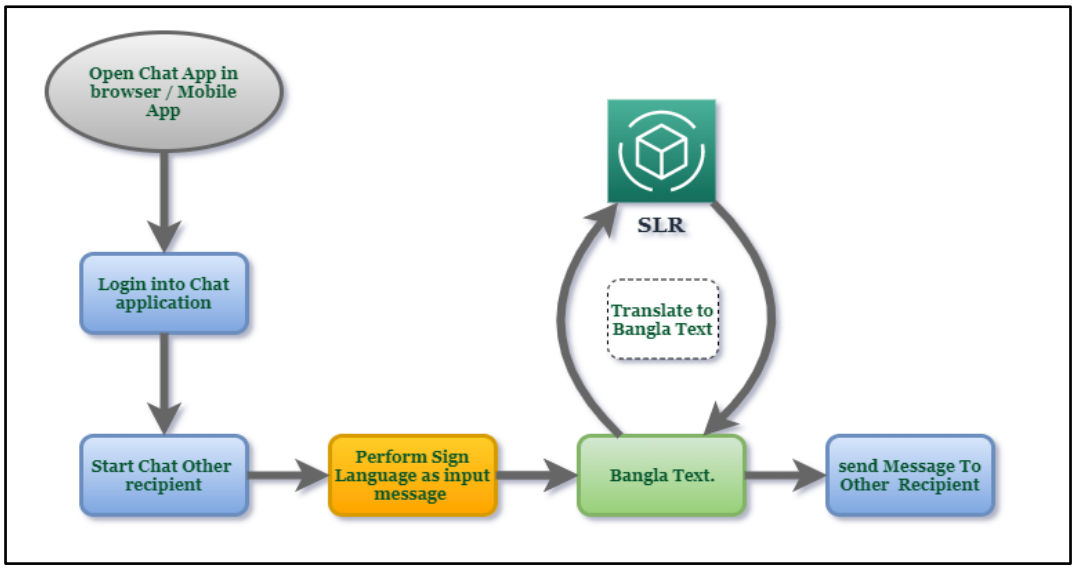


Figure 70. BdSL enabled web-based Chat Application

3.7.2 System Architecture (Chat Engine)

Application Architecture is the process of defining the framework of a system/solutions against business requirements. It involves the definition of the application landscape, aiming to optimize this landscape against the ideal blueprint. The Open Group's architectural framework, TOGAF ®, defines the objective of application architecture as "defining the major kinds of application systems necessary to process the data and support the business". Following figure shows a very high level architecture of our proposed chat application.

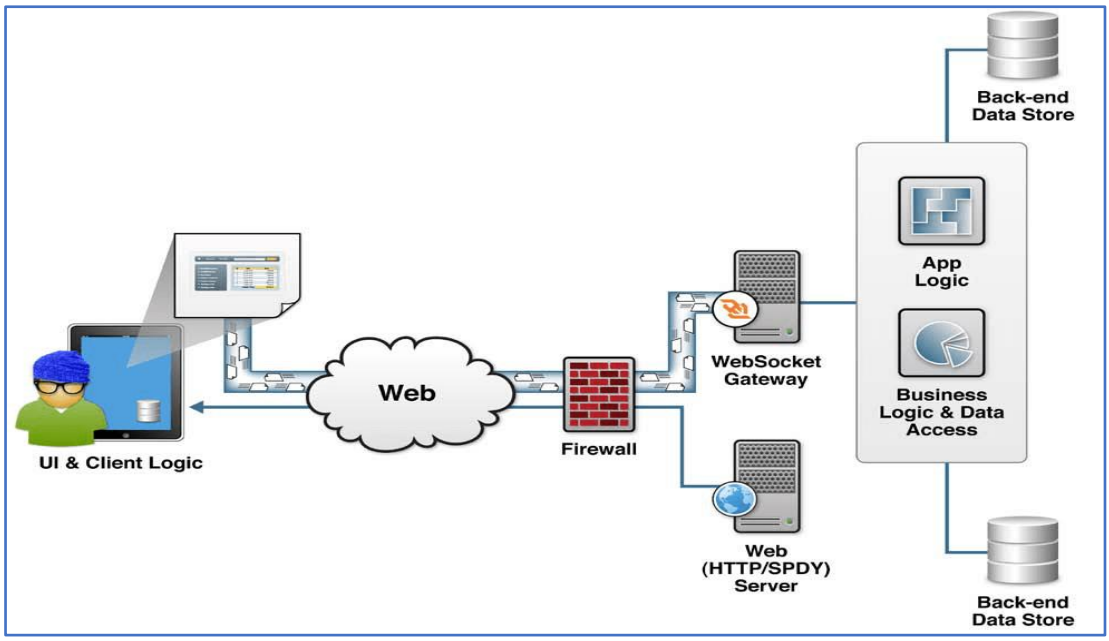


Figure 71. High Level Chat Application Architecture

If we dig down to the lower level of the architecture, then we can find that a chat consists of two major parts:

- Chat App or client part, which is a desktop, web, or smartphone chat application.
- Chat Server Engine or server part, which is a pool of external servers responsible for the chat operation. This is the place where all the chat magic happens.

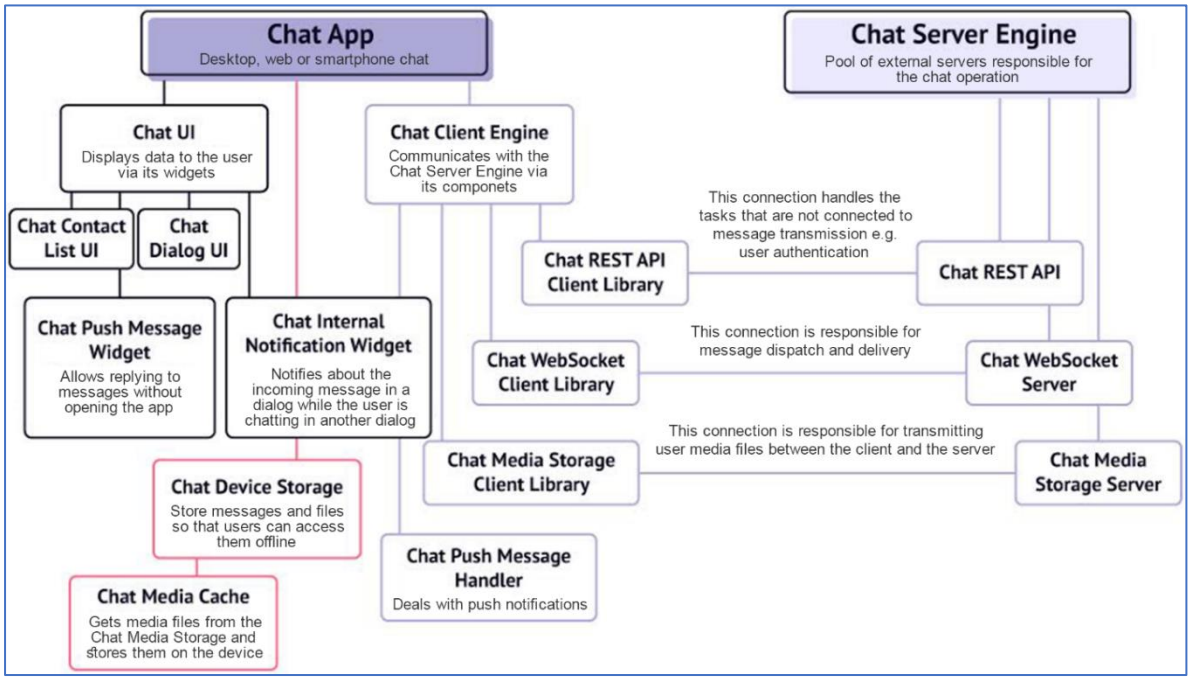


Figure 72. Low Level Application Architecture

Both parts contain various components that communicate to each other and bring the chat into action.

Chat Server Engine is a core of the chat architecture that handles message delivery and dispatch. In our version of chat architecture, it includes the following components:

- **Chat REST API** handles the tasks that are not connected directly to message dispatch and delivery, such as user authentication, changing of user settings, friend's invitation, downloading sticker packs, etc. The Chat App (the chat client part) communicates with the Chat REST API via the **Chat REST API Client Library**.
- **Chat WebSocket Server** is responsible for transmitting messages between users. The Chat App communicates with the Chat WebSocket Server via the **Chat WebSocket Client** Library. This connection is open two ways; that means users don't have to make requests to the server if there are any messages for them, they just get them right away.
- **Chat Media Storage Server** is a pool of servers responsible for storing user media files. The files are uploaded to them via the **Chat Media Storage Client Library**.

Chat App is the other major part of the chat architecture, the one that users directly interact with. It's split into three separate root components:

- **Chat Client Engine** handles all of the communication with the Chat Server Engine via its internal components: Chat REST API Client Library, Chat WebSocket Client Library and Chat Media Storage Client Library. It also comprises the **Chat Push Message Handler** that deals with push notifications.
- **Chat UI** displays data to users via its widgets: Chat Contact List UI, Chat Dialog UI, Chat Push Message Widget—extension for mobile apps that allow for replying to messages without opening the app and Chat Internal Notification Widget—a widget that pops up at the top of the screen while the user is chatting in a dialog and notifies about the incoming message in another dialog.
- **Chat Device Storage** is an internal database (read: device storage), which stores messages and files so that users can access them offline. Its internal component, Chat Media Cache, gets media files from the Chat Media Storage and stores them on the device so that the user can access them anytime without having to reach the Chat Media Storage every time.

Chat Application Architecture Discipline

As a discipline, application architecture lays the foundation for agility, scalability, and reliability in application landscape. Separation of components is at the core of our proposed chat application architecture. All the services that the architecture comprises are independent from each other to obey the foundational discipline.

- **Reusability:** When the chat components are designed separately, they can be easily reused in other projects.
- **Scalability:** Each architecture component is a potential bottleneck when a project grows bigger and begins to serve a larger audience. The advantage of independent

components is that they can be easily scaled separately without affecting the entire architecture.

- **Integration:** Independent components can be easily integrated with the customer's services upon request. For example, if the customer wants user files to be stored in their CRM, we can integrate the CRM with the Chat Media Storage without having to change the entire architecture.
- **Customization and migration:** All the architecture components can easily be adjusted to the customer's needs. We will build each of the components entirely from scratch if to perform specific tasks.

3.8 Web Application

As soon as SLR and T2SP engines are ready, exposed APIs from those engine could be used anywhere with proper authentication and authorization. Web based sign to text conversion system (SLR) will be in place utilizing that engine/API. The core system/solution interface which will be able to recognize Bangla Sign Language (BDSL code set) and convert the BDSL into Bangla Unicode text. The web interface will recognize sign language and relevant gestures from recorded and real-time motion image and will output sign Bangla to standard Bangla. The below screenshot shows signer performing signs (real time) and corresponding text will be generated:

বাংলা সাইন টু টেক্সট: The below interface comes from SLR web application. Here shows the feature, "বাংলা সাইন টু টেক্সট", where the text can be seen in the text field if a user shows the sign language through the web cam.

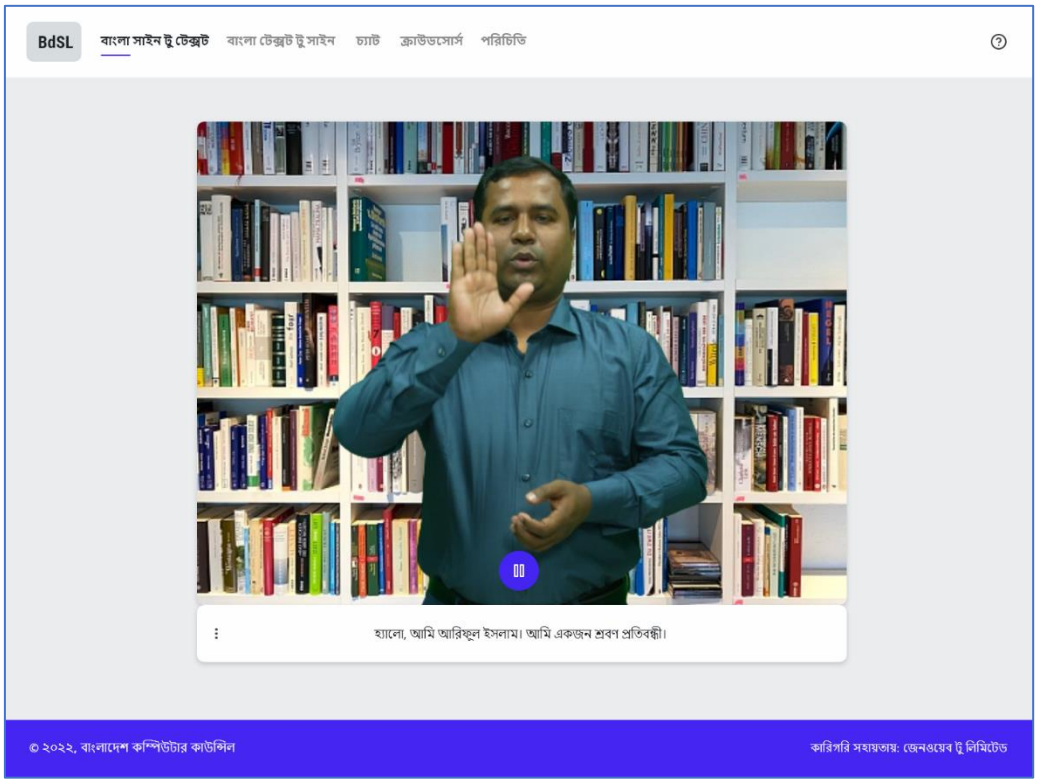


Figure 73. SLR in Web Application (sign to text)

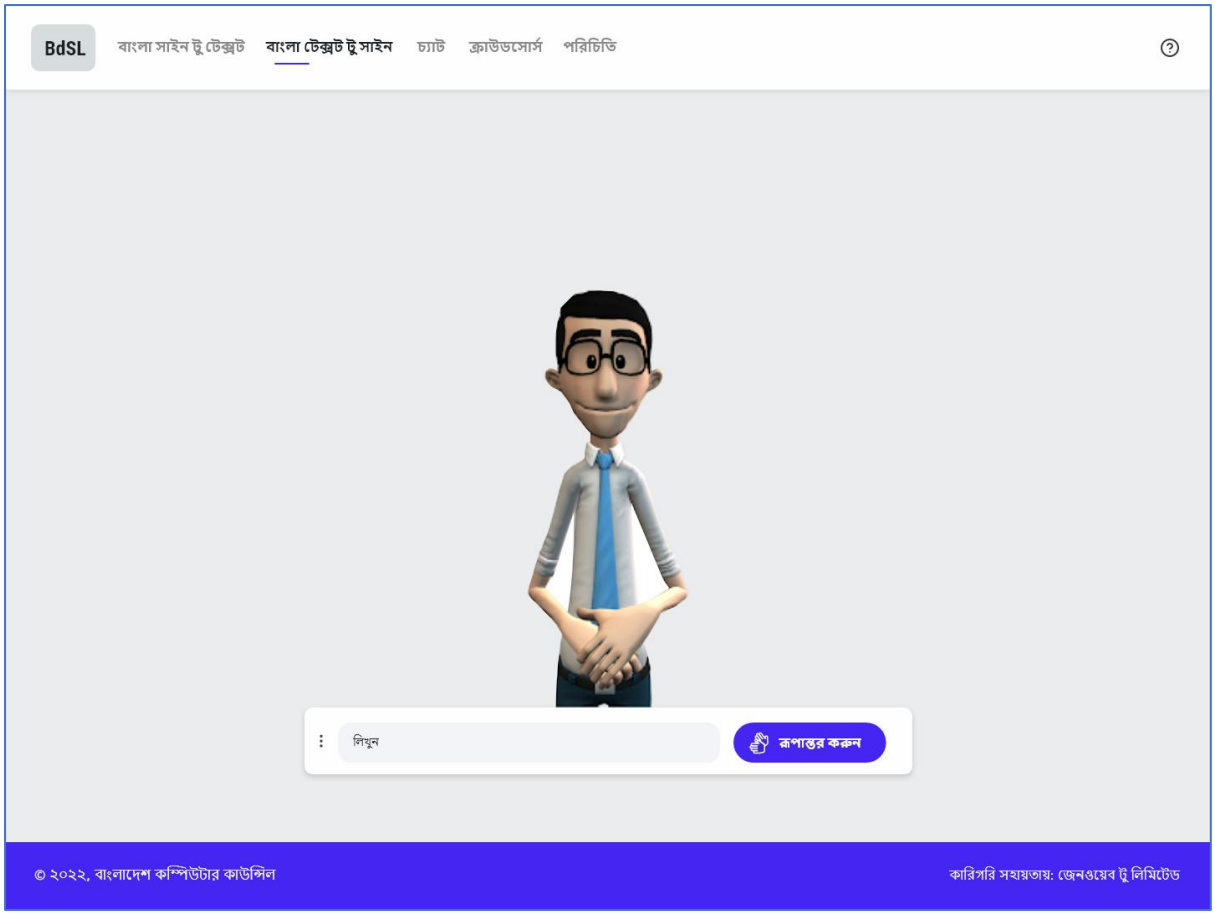


Figure 74. SLR in Web Application (Text to Sign)

An opposite way of SLR is text to sign puppet - the second part our SLR application brings a flavor of exchanging ideas from normal people to deaf people. The converter application will take Bangla text as inputs and will generate sign Bangla through cartoon/puppet (T2SP). The above screen show feeding input text will be acting by the cartoon.

চ্যাট (ইশারা ভাষা এবং লিখিত ভাষা): In below interfaces, a both side parallel communication can be seen in between a hearing disable user and a normal user. The below screen shows from the side of hearing disable user which is "চ্যাট (ইশারা ভাষা)" (Puppet) whereas the screen of normal user is "চ্যাট (লিখিত ভাষা)" (Text) on other side. Whatever written in text on other side, the written text convert into sign puppet and displays as animation. From this side when a hearing disable user replies, he/she can express the sign language in front of web camera which will convert into text and on the other side the normal user will get the corresponding text.

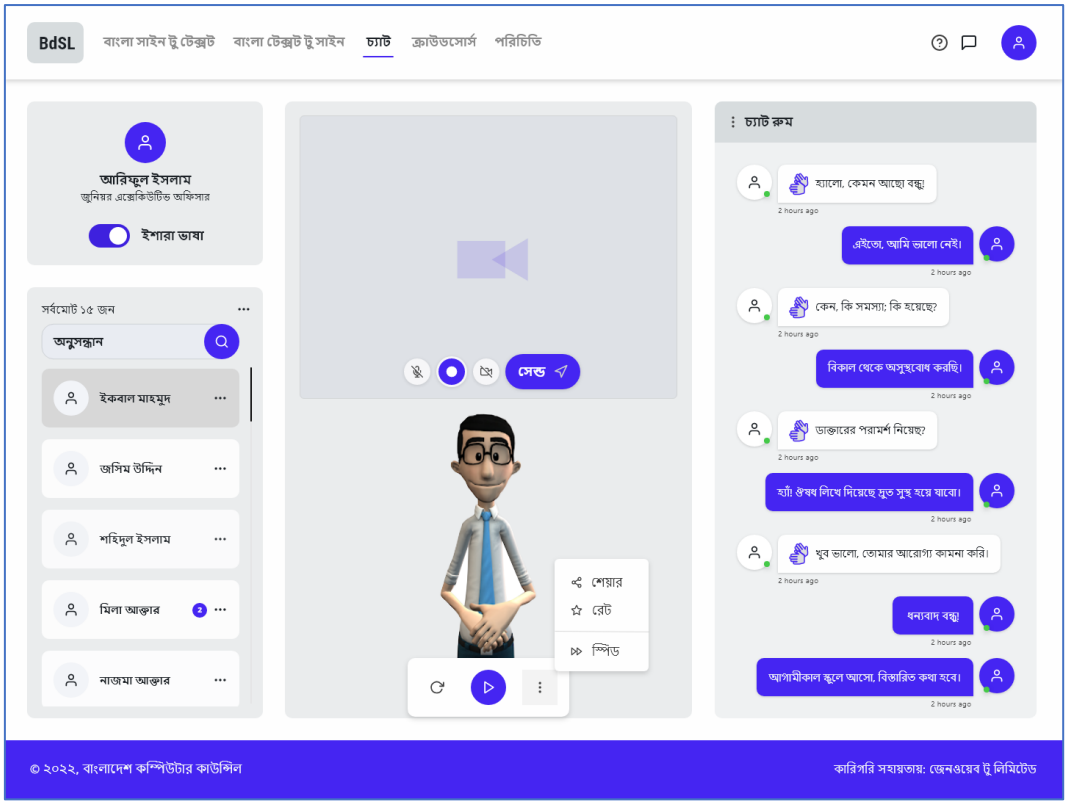


Figure 75. SLR Chatting in Web Chat (Puppet)

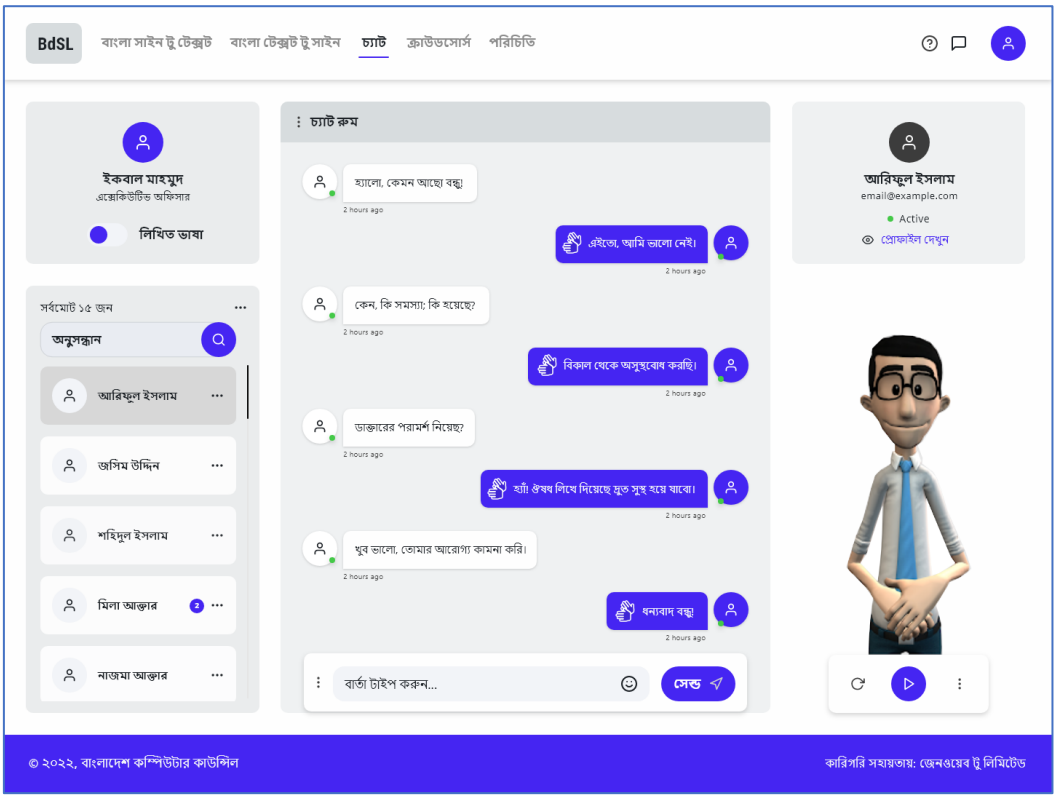


Figure 76. SLR Chatting in Web Chat (TEXT)

3.9 Desktop Application

3.9.1 Application Work Process

Along with the web application for the SLR, a desktop version will be built that will support Windows, Unix/Linux, and Mac operating system so that end user can enjoy the SLR's full advantage in any platform.

How the Desktop Application will work

Once built, the application will use the SLR API behind. As our SLR will be REST based, a mechanism needs to be implemented so that we can call/manipulate the SLR API via desktop application.

Authentication and Authorization

Role-based authentication and authorization will be implemented so that only authorized user can access the desktop application from any computer.

Real-Time Video Recording

When the application will run in the desktop, it's considered that the computer will be equipped with a camera/webcam that will record video of signer. The captured video will be sent to the server with a configuration rate.

Real-Time Video Processing

This real time recorded video will keep sending to the server using SLR API. While serve receives the continuous images, it starts processing and replies back with Bengali sign language. There will be lot of configurations/settings in the desktop application like

- Capture video with optimized resolution
- FPS (Frame Per Second) – it has quality measures
- The number of images will be sent to the server per second.
- Log the error when application crashes/doesn't response.
- Notify users when there is low bandwidth etc.

Sending image from the captured video continuously from desktop application to the sever has several different bottlenecks like accuracy, performance and low bandwidth. Image sending rate might become a problem while sending to the server, so a fine-tuned rate should be chosen so that expected result comes in. The application will be equipped to keep the balance network request speeds with computation speeds of neural network (to convert the sign to language).

So, to the core, our desktop application will capture video and sends images from the video to the server sequentially. Each time, the server classifies the image and presents probabilities for each letter/sign. It keeps a running cache of classified images. When it feels confident about the sign being made by the user, it records the top-5 most likely letters/signs based on the cache. It then clears the cache, and lets the user know to move on to the next letter/sign if configured such way.

Once the user has indicated that they're finished signing, the top-5 letters for each position in the spelled word are passed to a custom unigram language model based on the Corpus. The model takes into account substitution costs for similar-looking letters that are often confused by the classifier as well as the probabilities for the top-5 letters at each position in the word. Using these, single word unigram probabilities and a handful of other heuristics, it returns the optimal word to the user and hence, on the other side, other user sees translated text/sentences.

T2SP for Desktop Application

Text to Sign puppet application for Desktop version will have all the functionalities similar to web version. It will be able to render sing language animation via a puppet from Bangla text. User will get the option to render text as it is or avatar from the provided input.

Sign Language Communicator for Chat in Desktop Application

Like in the web Application, user will be able to enjoy full-fledge chat capabilities in the Desktop Application also. Features and functionalities of BdSL enabled desktop-based Chat Communicator.

- User will be able to register in this chat application by creating their profile.
- User will be able to send Instant messages to other recipients considering conventional messaging is text to text between sender and receiver, here it is Sign Language to Text.
- User will be able to create an unlimited number of multiple groups with other recipients.
- They will also be able to invite people to join in chat groups etc.
- System will be able to handle instant and concurrent messaging to all the members of the group.
- During chat user will get text and sign language as input options. User may type a text message or perform sign language for input of chat. System will decide the messaging mode upon user selection of input mode. It will work just like English and Bangla keypad change for input.
- User can search for other users using various information like name, email, mobile number, etc. Search result will be shown as list.
- User will get notifications of incoming chats. It a user is offline or online, then status will be detected automatically by the system.
- Offline chat feature for both one to one and group chat will be available.
- Option will be there to save the chat session depending on the captured video of sign language.

Note: A common platform application building mechanism is under investigation using which we can server both web and desktop applications - if successful result comes in, this technique will be followed so that with the single source of application both web and desktop can be served.

3.9.2 User & Resource Accessibility

Advance Settings/Configuration

As mentioned above, the application will have master configuration settings so that managing features in the application becomes easier. To control the application and to get the better performance we will adapt as much as configuration parapets as possible. A great advantage of configuration-based parameter will be the application won't need much change and redeployment.

Primarily Proposed Configuration Parameters:

SL	Parameter Name	Parameter Value
1	Aspect Ratio	[16:9]
2	Resolution	[defined value]
3	Frame rate	[30]
4	Key frame	[2 seconds]
5	Bitrate (video)	[20–51 Mbps]
6	Bitrate encoding	[CBR]
7	Pixel aspect ratio	[Square]
8	Channel layout	[Stereo]
9	Protocol	[RTMP/RTMP]
10	Capture video with optimized resolution	[TRUE]
11	The number of images will be sent to the server per second.	[TRUE]
12	Notify users when there is low bandwidth	[TRUE]

Table 14. Proposed Configuration Parameters

Note: All the configuration values are sample only, in real application these will change and subject to add remove.

Error Handling

In software world error is uncertain and handling that error in real-time application is cumbersome. So, to minimize the downtime for the error application will be given intelligent in that way. Below are some techniques using which we can minimize the downtime:

- Write error log to the DB/OS designated folder.
- Send error via email to the designated person (email address will be configured)
- Instant notification – if can be configured SMS/WhatsApp.

3.10 Approach for Crowdsourced Data Collection

3.10.1 The Challenge

Data collection is a major bottleneck in machine learning and an active research topic in multiple communities. There are largely two reasons data collection has recently become a critical issue. First, as machine learning is becoming more widely used, we are seeing new applications that do not necessarily have enough labeled data. Second, unlike traditional machine learning where feature engineering is the bottleneck, deep learning techniques automatically generate features, but instead require large amounts of labeled data. Interestingly, recent research in data collection comes not only from the machine learning, natural language, and computer vision communities, but also from the data management community due to the importance of handling large amounts of data.

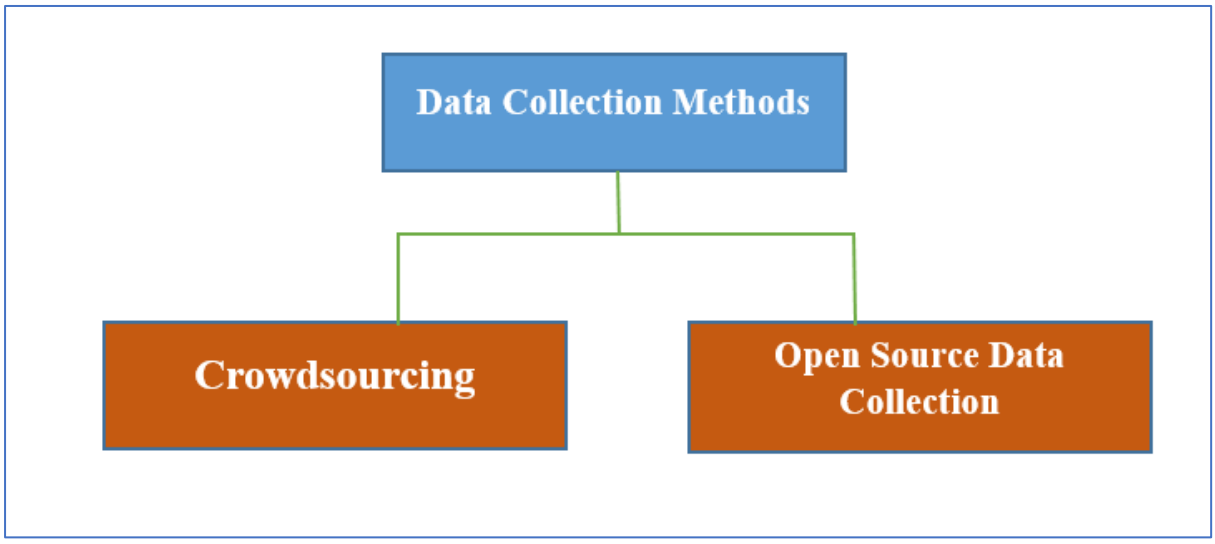


Figure 77. Crowdsourced Data Collection Method

3.10.2 Collecting Data

Sign languages are used by the deaf and mute community of the world. These are gesture based languages where the subjects use hands and facial expressions to perform different gestures. Furthermore, like natural languages, there exist different dialects for many sign languages. In order to facilitate the deaf community several different repositories of video gestures need to be make available so that they can uploaded their own gestures.

As a principal contribution, this research will presents a framework for building a parallel corpus for sign languages by harnessing the powers of crowdsourcing with editorial manager, thus it engages a diversified set of stakeholders for building and validating a repository in a quality controlled ways. Rules need to be imposed so that anybody can't upload anything and at the same time admin should be super active to align the data with the SLR model and need to train them accordingly. Multiple independent repo will have to be managed so that live/UAT database doesn't get affected.

3.10.3 Crowdsourced Data Validation

In order to effectively infer correct labels from noisy crowdsourced annotations, learning-from-crowds models have introduced expert validation. However, a huge concern needs to be paid on facilitating the validation procedure. We propose an interactive method to assist experts in verifying uncertain instance labels and unreliable workers. Given the instance labels and worker reliability inferred from a learning-from-crowds model, candidate instances and workers are selected for expert validation. The influence of verified results is propagated to relevant instances and workers through the learning-from-crowds model.

To facilitate the validation of annotations, a visualization method should be there to indicate the confusing classes for further exploration, a constrained projection method should also be there to show the uncertain labels in context, and a scatter-plot-based visualization should be there to illustrate worker reliability. The three visualizations are tightly integrated with the learning-from-crowds model to provide an iterative and progressive environment for data validation. However, a decision-making flow chart should be there to scrutinize while gathering data from anywhere including crowdsourcing.

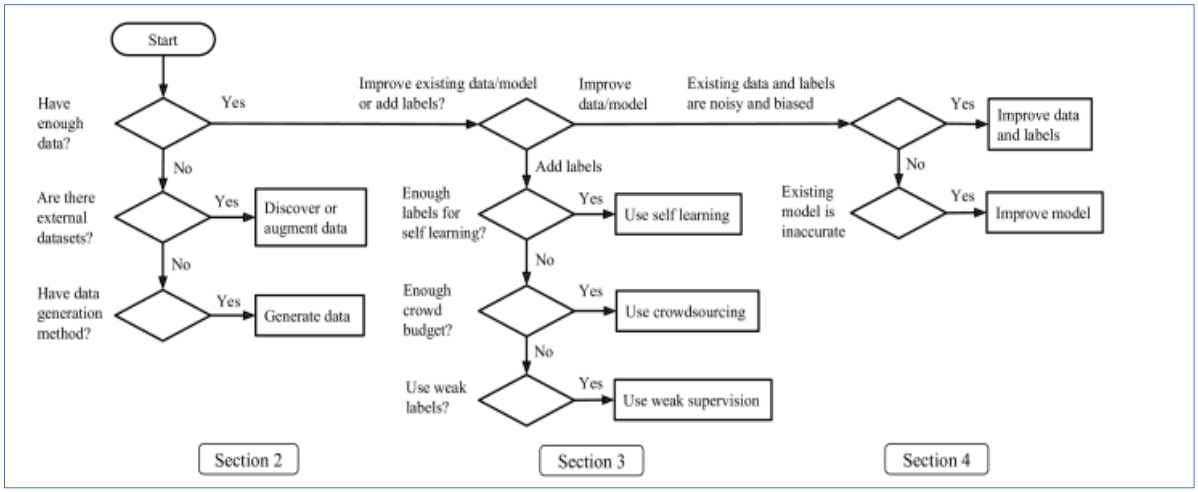


Figure 78. A decision flow chart for data collection

A cloud-based admin portal will be implemented for crowdsourcing data for BdSLR so that training dataset can be augmented to clean accuracy. A monitoring and validation approach will be implemented so that garbage can't be inputted to ruin the existing proven model.

3.11 Use Case Specification

3.11.1 Sign Language Recognition (SLR)

Use Case Name	Recording Video
Use Case ID	UC-01
Primary Actor	Signer
Secondary Actor	Camera
Objective	Track finger, hand movement & facial gesture
Prediction	The signer uses standardized signs, specification of camera to capture the signs
Output	Video of sentence & word

Use Case Name	Conversion of Sentence gloss from video data
Use Case ID	UC-02
Primary Actor	Camera
Secondary Actor	Sentence model (tokenizer)
Objective	To break a passage in separate sentences
Prediction	Sentence model will identify stop word from video segment and separate each sentence.
Output	Sentence or sentence gloss

Use Case Name	Tokenization of sentence to word
Use Case ID	UC-03
Primary Actor	Sentence model (tokenizer)
Secondary Actor	Word model (tokenizer)
Objective	To break a sentence into multiple words or word gloss
Prediction	Word model identifies the word gloss from sentence and dispersed them to word gloss
Output	Word or word gloss

Use Case Name	Annotation of video data
Use Case ID	UC-04
Primary Actors	Sentence model, word model
Secondary Actor	Annotation tool
Objective	To get posture keypoint
Prediction	Labeled the video with word gloss
Output	Posture keypoint, coordinate and timeline

Use Case Name	Coordination of hand gesture
Use Case ID	UC-05
Primary Actor	Annotation tool
Secondary Actor	Keypoint generation model
Objective	To get specific keypoint
Prediction	Model will identify the coordinate of hand
Output	Keypoint coordinate

Use Case Name	Coordination of facemesh
Use Case ID	UC-06
Primary Actor	Annotation tool
Secondary Actor	FaceMesh model
Objective	To get face mesh
Prediction	Model will point the face mesh
Output	Face sign label

Use Case Name	Sign performer detection
Use Case ID	UC-07
Primary Actor	Annotation tool
Secondary Actor	Detector model
Objective	To validate the sign performer (human person)
Prediction	Model detects that the sign performer is a human or not
Output	Returns binary classifier (positive or negative)

Use Case Name	Transformation of word gloss to keypoint
Use Case ID	UC-08
Primary Actors	Keypoint generation model, face mesh model, detector model
Secondary Actor	Sign (keypoint) to word model,
Objective	To get unique word
Prediction	Feature matrix will transform to word gloss
Output	Word gloss

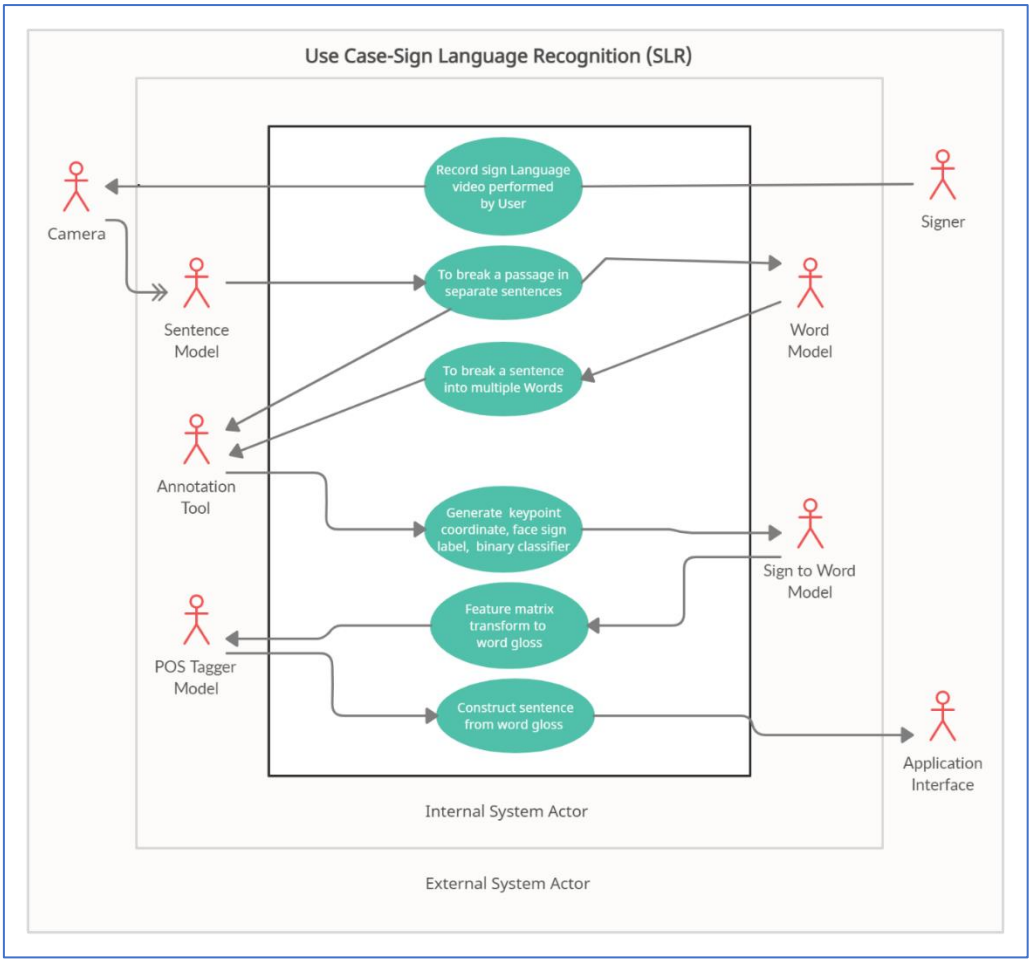
Use Case Name	Construction of sentence from word
Use Case ID	UC-09
Primary Actor	Sign (keypoint) to word model
Secondary Actor	Sentence(POS Tagger) model
Objective	To get an unorganized sentence
Prediction	Word model will process all the unique word or word gloss and form a sentence.
Output	Unstructured sentence

Use Case Name	Organizing sentence and visualization
Use Case ID	UC-10
Primary Actor	Sentence(POS Tagger) model
Secondary Actor	Application
Objective	To view normal text
Prediction	Sentence model organize the unstructured sentence and visualize the text
Output	Amorphous sentence become normal text and shows on application interface

Use Case Name	Collection of sign language running images
Use Case ID	UC-11
Primary Actor	Signer
Secondary Actor	System
Pre-conditions	The device corresponding camera must be powered on.
Basic flow	• Signer will perform Bangla sign language • System captures the running image through camera
Alternate flow	• If camera is not powered on, system can't capture running image • System can't connect with signer device if there is no internet connection
Post-conditions	The system will process the running images for sign to text conversion

Use Case Name	Conversion of sign language
Use Case ID	UC-12
Primary Actor	System
Secondary Actor	Regular user
Pre-conditions	The system should be running
Basic flow	• System will process all the running images of Bangla sign language • Converts the sign language into Bangla text • Display the Bangla text to user
Alternate flow	• Returns nothing if signer didn't show any sign language • Can't able to display text if there is no internet connection with user device
Exceptional Flow	• System shows exception if signer shows gesture which does not match with Bangla sign language
Post-conditions	System will display the Bangla text corresponding to sign language

Use Case Name	Suggestion/Correction input as feedback & verification
Use Case ID	UC-13
Primary Actor	User
Secondary Actor	System
Pre-conditions	The system displayed converted Bangla text which needs correction
Basic flow	• User makes correction in the text as feedback • System will pass the correction feedback for verification
Alternate flow	• User didn't make any correction
Post-conditions	System asks to verify the correction to admin



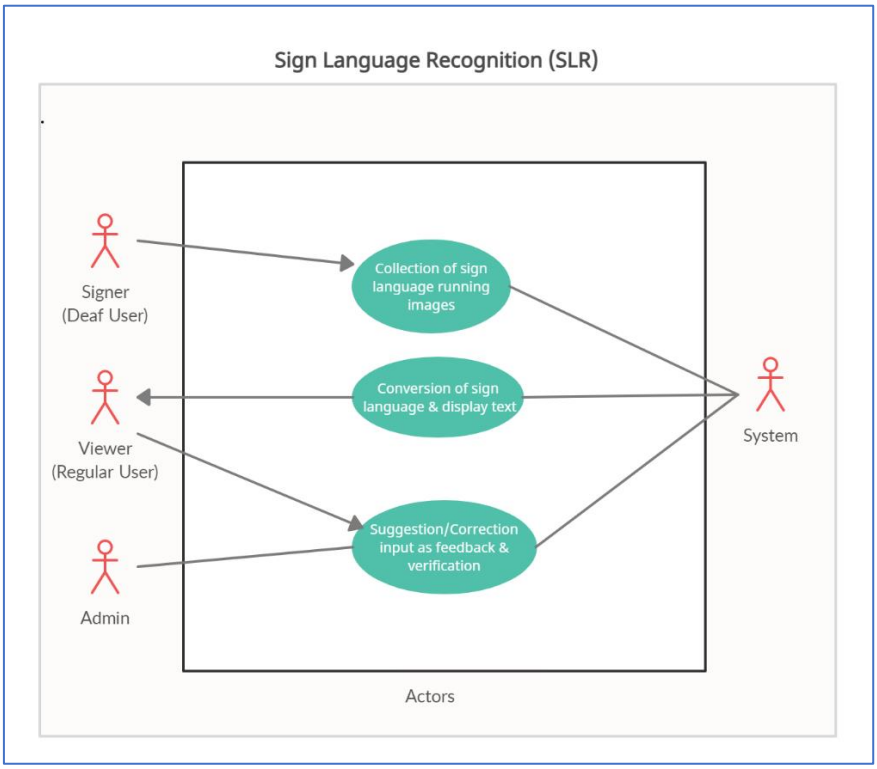


Figure 79. Use Case: Sign Language Recognition (SLR)

3.11.2 Text to Sign Puppet (T2SP)

Use Case Name	Processing of sentence
Use Case ID	UC-01
Primary Actor	User (Through application interface)
Secondary Actor	Sentence model (tokenizer)
Objective	To break a passage in separate sentences
Prediction	Sentence model will identify stop word from normal text input by user and separate each sentence.
Output	Sentence or sentence gloss

Use Case Name	Tokenization of sentence to word
Use Case ID	UC-02
Primary Actor	Sentence model (tokenizer)
Secondary Actor	Word model (tokenizer)
Objective	To break a sentence into multiple words or word gloss
Prediction	Word model identifies the word gloss from sentence and dispersed them to word gloss
Output	Word list or word gloss list

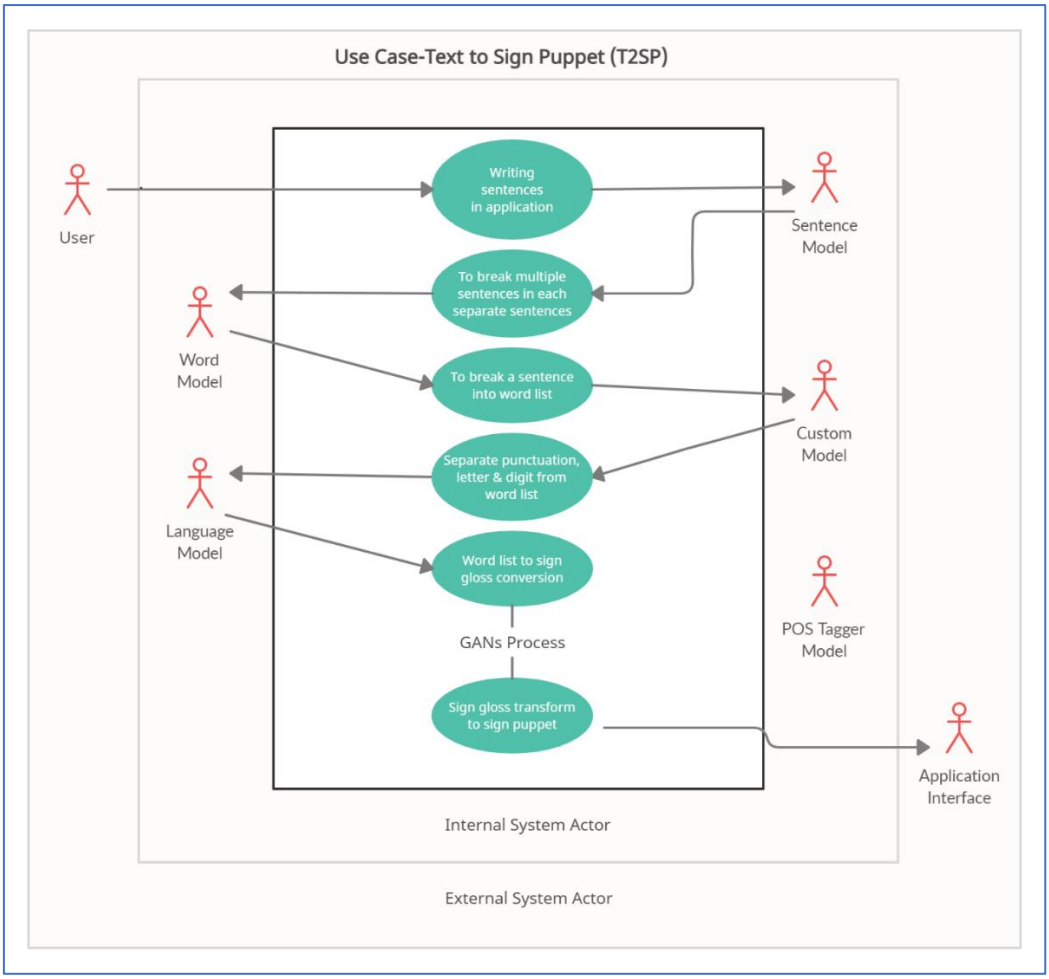
Use Case Name	Text cleaning & preprocessing
Use Case ID	UC-03
Primary Actor	Word model (tokenizer)
Secondary Actors	Custom models (punctuation, letter, digit)
Objective	To separate punctuation, letter, digit from the word list
Prediction	Model will discrete any punctuation, single letter and digit from the word list and keep them in separate list.
Output	Clean word list

Use Case Name	Conversion of word list
Use Case ID	UC-04
Primary Actors	Custom models (punctuation, letter, digit)
Secondary Actor	Language Model
Objective	To get sign gloss
Prediction	Language model converts the word list into multiple sign gloss
Output	Sign gloss

Use Case Name	Transformation of sign puppet
Use Case ID	UC-05
Primary Actor	Custom language Model
Secondary Actor	Animated Interface
Objective	To visualize sign puppet, performing sign language
Prediction	Through the generative modeling (GANs) process sign glosses connects with dataset and transform into sign puppet
Output	Sign puppet, acting sign language

Use Case Name	Writing Bangla text
Use Case ID	UC-6
Primary Actor	Regular User
Secondary Actor	System
Pre-conditions	The device should be connected with internet connection
Basic flow	• User will write Bangla text • System will receive the text for processing sign puppet
Alternate flow	• If text is written in a language other than Bangla, the system will not understand the language. • System can't able to receive the written text if there is no internet connection
Post-conditions	The system will process the Bangla text to sign puppet.

Use Case Name	Convert Bangla text to sign puppet animation
Use Case ID	UC-7
Primary Actor	System
Secondary Actor	Hearing disabled user
Pre-conditions	The system should be running
Basic flow	• System will process all the Bangla text written by regular user • Converts the text into animated sign puppet • Display the sign puppet animation to hearing disabled user
Alternate flow	• Display nothing if system doesn't get any text to convert • Can't able to display sign puppet animation if there is no internet connection with user device
Exceptional Flow	• System shows exception if the written text has mixing of other language with Bangla language.
Post-conditions	System will display the sign puppet animation corresponding to Bangla text.



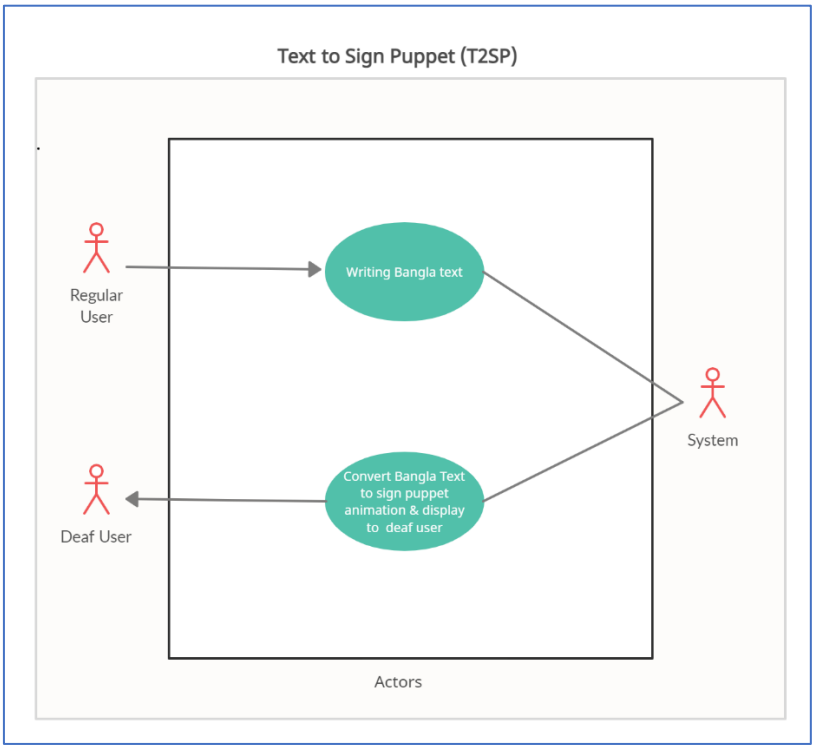


Figure 80. Use Case: Text to Sign Puppet (T2SP)

3.11.3 Bangla to Braille Converter with Editor (B2BC)

Use Case Name	Parsing of electronic Bangla text
Use Case ID	UC-01
Primary Actor	User
Secondary Actor	Input Module
Objective	To read electronic Bangla text from the archive(Microsoft Office Word (doc, docx) or any text file format)
Prediction	Module will parse the Bangla text from different file formats and pass it to input buffer
Output	Bangla Unicode

Use Case Name	Processing of Bangla Unicode text
Use Case ID	UC-02
Primary Actor	Input Module
Secondary Actor	Input buffer
Objective	To process the Unicode text to single character
Prediction	Sends character of Unicode Bangla text one by one to rule engine.
Output	Character

Use Case Name	Construction of braille rules
Use Case ID	UC-03
Primary Actor	Input buffer
Secondary Actors	Rule engine
Objective	To translation of the Bengali Unicode text into Braille rules
Prediction	Reads the character of input buffering module and by translation rule, it determines whether a sequence of character leads to a token or not. If forms a token, sent to the translator.
Output	Bangla token and rule number

Use Case Name	Translation of Bangla character to braille symbol
Use Case ID	UC-04
Primary Actors	Rule engine
Secondary Actor	Translator
Objective	Mapping Bangla character to braille symbol with rule table (where all grammatical rules placed)
Prediction	Translate Bangla character to Braille symbol with the appropriate grammatical rules
Output	Braille token

Use Case Name	Maintaining of braille tokens
Use Case ID	UC-05
Primary Actor	Translator
Secondary Actor	Output buffer
Objective	To maintain braille symbols with token
Prediction	Receive the translated braille tokens from the translator module then stores in memory and asks translator for next token
Output	Braille symbols

Use Case Name	Generating the braille document
Use Case ID	UC-06
Primary Actor	Output buffer
Secondary Actor	Output module
Objective	To generate the translated braille output
Prediction	Collects all translated braille Symbols from output buffer and generate the translated braille output into braille document based on user's choice.
Output	Braille document

Use Case Name	Printing of braille document
Use Case ID	UC-07
Primary Actor	Output module
Secondary Actor	Printer
Objective	To print braille document
Prediction	Printing braille document through braille printer which will be with braille embossers.
Output	Printed document of braille code

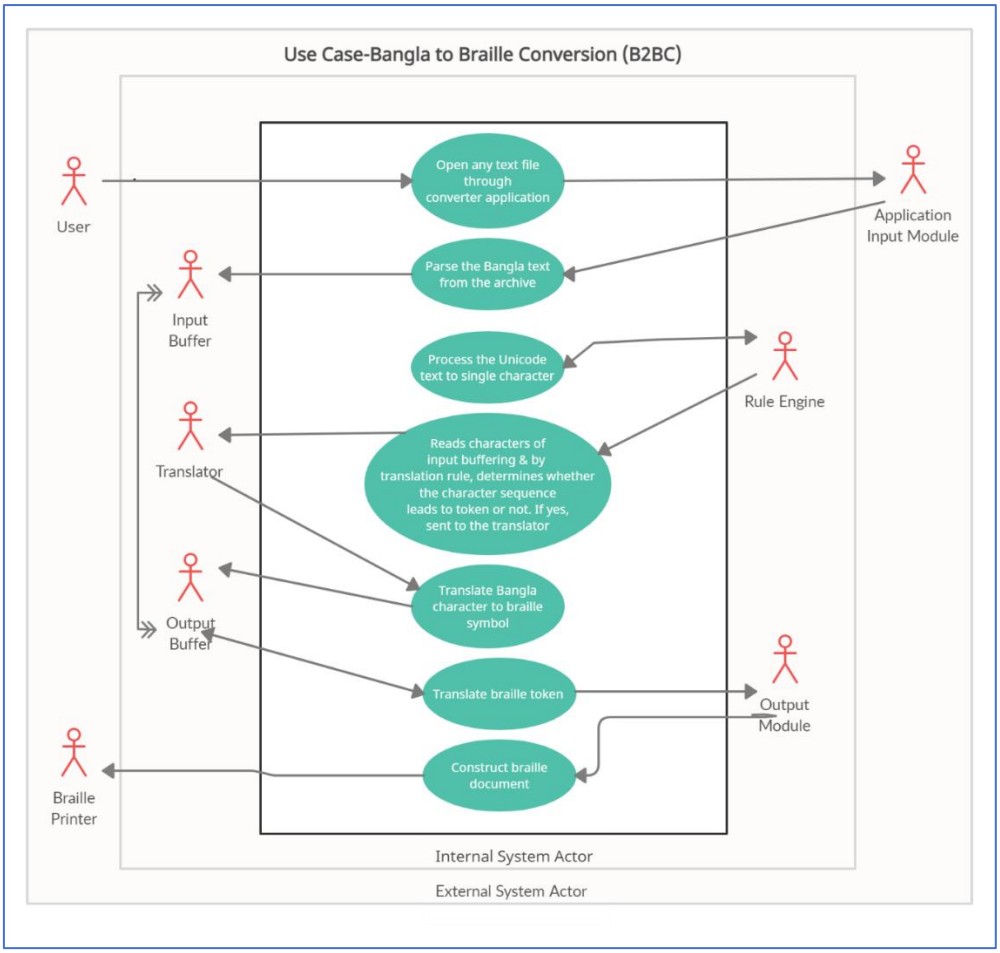
Use Case Name	Open text file in braille converter text editor
Use Case ID	UC-8
Primary Actor	User
Secondary Actor	Converter text editor
Pre-conditions	• Converter application should open without error • Upload button should work fine
Basic flow	• Upload any Bangla text file • System opens the Bangla text file in converter text editor
Alternate flow	• If text file doesn't contain any written Bangla text, there will be no content in the converter • Fails to upload the text file if button doesn't work
Post-conditions	System opens the Bangla text file with conversion button

Use Case Name	Edit text in braille converter text editor
Use Case ID	UC-9
Primary Actor	User
Secondary Actor	Converter text editor
Pre-conditions	• Text editor should have the feature to edit content
Basic flow	• Edit the existing Bangla text content
Alternate flow	• Text editor can't able to show exact text format if editing Bangla text doesn't have same Unicode format.
Post-conditions	User can edit the existing content in Braille converter text editor

Use Case Name	Convert Bangla text to braille code
Use Case ID	UC-10
Primary Actor	User
Secondary Actor	Braille converter
Pre-conditions	Converter system should be able to read Bangla text content
Basic flow	• User successfully click on the convert button • System will read the Bangla text content from text editor • Convert the Bangla text content into braille code
Alternate flow	• If there is no text content in editor, system will have nothing to read. • Fails to convert if the text in editor contains any other language other than Bangla.
Post-conditions	System converts the Bangla text content into a braille code

Use Case Name	Edit braille code in braille editor
Use Case ID	UC-11
Primary Actor	User
Secondary Actor	Braille editor
Pre-conditions	• Edit button for Bangla braille edit button should work fine • Braille editor should have the feature to edit braille code
Basic flow	• User successfully click on the edit button • A virtual keyboard should onboard which will contain all braille code • Edit the braille code wherever necessary in braille editor
Alternate flow	• User can't able to edit braille code if edit button doesn't work • Fails to edit braille code if the virtual keyboard doesn't onboard.
Post-conditions	User can edit the converted braille code successfully

Use Case Name	Print of braille document
Use Case ID	UC-12
Primary Actor	User
Secondary Actor	Braille printer
Pre-conditions	User should be able to print braille code through braille printer
Basic flow	• System connects with braille printer through user's printing instruction • Braille printer prints the braille code with embossers
Alternate flow	• Converter can't able to connect with braille printer if printer cable is unplugged
Post-conditions	Braille printer will print the braille document contains braille code



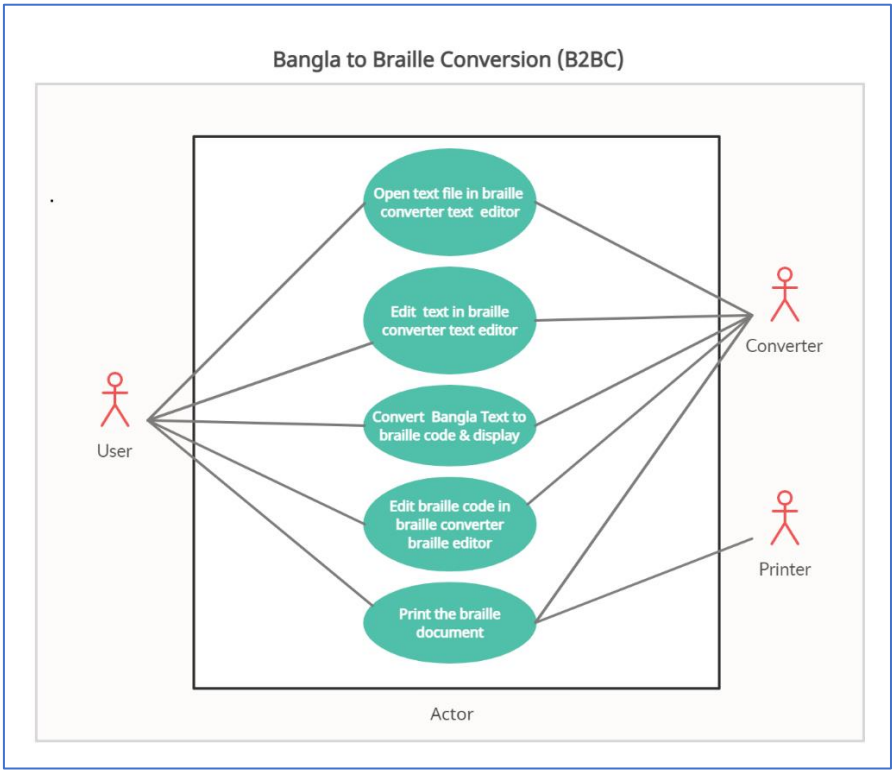


Figure 81. Use Case: Bangla to Braille Converter (B2BC)

4. Development Methodology

Based on several different components, a varied software development strategy will be followed for SD-17. With the touch of modern trends, modern development methodologies will be followed as much as possible. For traditional software (Web SLR, Braille Converter, Crowdsourced data collection, Back office admin portal) traditional software development methodologies will be followed whereas for AI/ML based application (SLR engine, T2SP engine) machine learning development lifecycle will be followed. Finally for mobile apps development - a design first approach will be catered in conjunction with traditional software development lifecycle (SDLC).

4.1 Machine Learning Development Life Cycle (for SLR and T2SP)

Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Following are the major steps that are involved in the machine learning life cycle:

- Gathering Data
- Data Preparation
- Data Analysis
- Model Training
- Model Testing
- Deployment
- Monitoring and Optimization

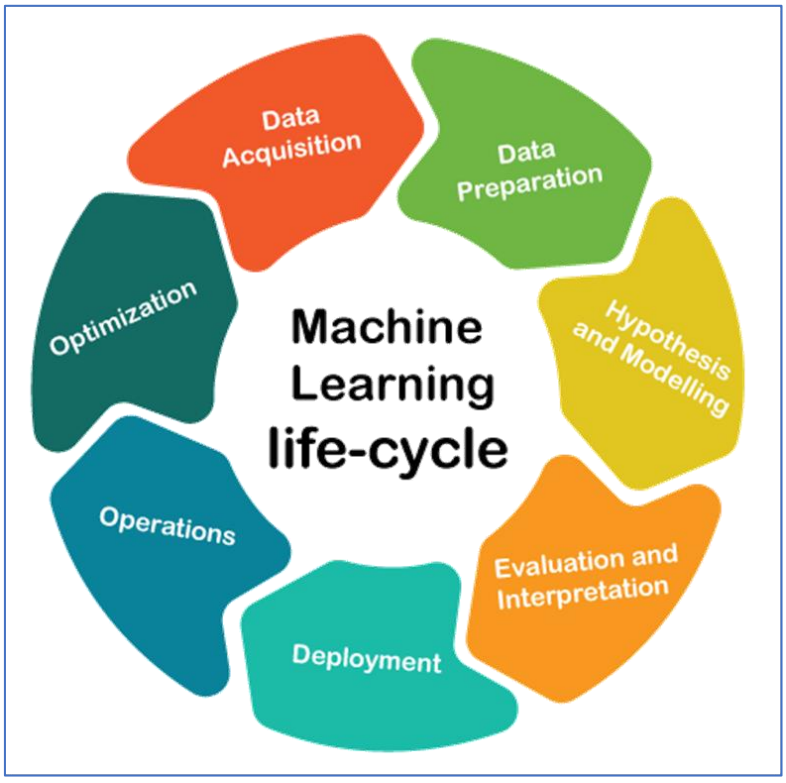


Figure 82. Machine Learning Life Cycle (MLLC)

The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

Gathering Data

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **internal collection**, **crowdsourcing**, **internet**, or **IOT devices**, but we must acquire appropriate data to solve one particular problem. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- Identify various data sources
- Collect data
- Integrate the data obtained from different sources

It does not require any data scientist to gather the data. Anyone who has prior knowledge regarding the actual difference between the several data sets that are freely available and who knows how to make hard-hitting decisions about any organization's investment strategy regarding collecting data will serve best for this role. By performing the above task, we get a coherent set of data, also called as a **dataset**. It will be used in further steps.

Data Preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training. This may be the most tedious and time-consuming task in this cycle, which involves identifying various data quality issues. Usually, whenever the data is acquired, it cannot be analyzed as it might comprehend misplaced entries, irregularities and semantic errors. Thus, to use such kind of data, the data scientists reformat and clean the data manually by either editing it in the spreadsheet or simply by scripting the code. This is the phase where we split the data into training, validation, and test sets. Another name of the data preparation step is the data cleaning or data wrangling phase.

This step can be further divided into two processes:

- **Data exploration**: It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data. A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.

- **Data pre-processing:** Now the next step is preprocessing of data for its analysis.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- Missing Values
- Duplicate data
- Invalid data
- Noise

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. Here, we perform Exploratory Data Analysis (EDA) to understand the available data for building the ML model. This step involves:

- Selection of analytical techniques
- Building models
- Review the result

This process leads to the following:

- Understanding the data schema and characteristics that are expected by the model.
- Identifying the data preparation and feature engineering that are needed for the model.

Model Training

The data scientist implements different algorithms with the prepared data to train various ML models. In addition, we subject the implemented algorithms to hyperparameter tuning to get the best performing ML model. The output of this step is a trained model.

Model Testing

After training the model on data, we evaluate the end result to see how well it performed or how reliable it is in real-life situations. The output of this step is a set of metrics to assess the quality of the model. Each performance metric has a distinct evaluation metrics. For instance, if we want our machine learning model to predict the daily stock, then it is highly recommended to consider the RMSE (root mean squared error) for the evaluation. Else, we consider performance metrics like average accuracy, AUC and log loss order for classifying spam emails. Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

Deployment

The validated model is deployed to a target environment to serve predictions. The term deployment can be defined as an application of a model that makes the prediction by means of

the new data. Building a model is generally not the end of the project. Even though the purpose of the model is to intensify the data's knowledge, the gained knowledge has to be organized and presented in such a way that it can be easily used by the customer. This deployment can be one of the following:

- Microservices with a REST API to serve online predictions.
- An embedded model to an edge or mobile device.
- Part of a batch prediction system.

Monitoring and Optimization

The model predictive performance is monitored to potentially invoke a new iteration in the ML process. Basically, in this phase, it monitors the performance of the model in terms of upgrade and downgrade. Data scientists take help from an individual data science project for shared learning and to boost up the implementation of alike data science projects in the coming future. It reinstructs the machine learning model in production when new data sources are coming in or take necessary steps for upgrading the performance of the machine learning model.

MLOps - Continuous Integration, Continuous Delivery, and Continuous Training

To develop and operate complex systems, we can apply DevOps principles to ML systems (MLOps). ML and other software systems are similar in continuous integration of source control, unit testing, integration testing, and continuous delivery of the software module or the package. However, in ML, there are a few notable differences:

- Continuous Integration (CI) is no longer only about testing and validating code and components, but also testing and validating data, data schemas, and models.
- Continuous Delivery (CD) is no longer about a single software package or a service, but a system (an ML training pipeline) that should automatically deploy another service (model prediction service).
- Continuous Training (CT) is a new property, unique to ML systems, that's concerned with automatically retraining and serving the models.

For a rapid and reliable update of the pipelines in production, we need a robust automated CI/CD system. This automated CI/CD system lets our data scientists rapidly explore new ideas around feature engineering, model architecture, and hyperparameters. They can implement these ideas and automatically build, test, and deploy the new pipeline components to the target environment.

The following diagram shows the implementation of the ML pipeline using CI/CD, which has the characteristics of the automated ML pipelines setup plus the automated CI/CD routines.

For a rapid and reliable update of the pipelines in production, we need a robust automated CI/CD system. This automated CI/CD system lets our data scientists rapidly explore new ideas around feature engineering, model architecture, and hyperparameters. They can implement these ideas and automatically build, test, and deploy the new pipeline components to the target environment.

The following diagram shows the implementation of the ML pipeline using CI/CD, which has the characteristics of the automated ML pipelines setup plus the automated CI/CD routines.

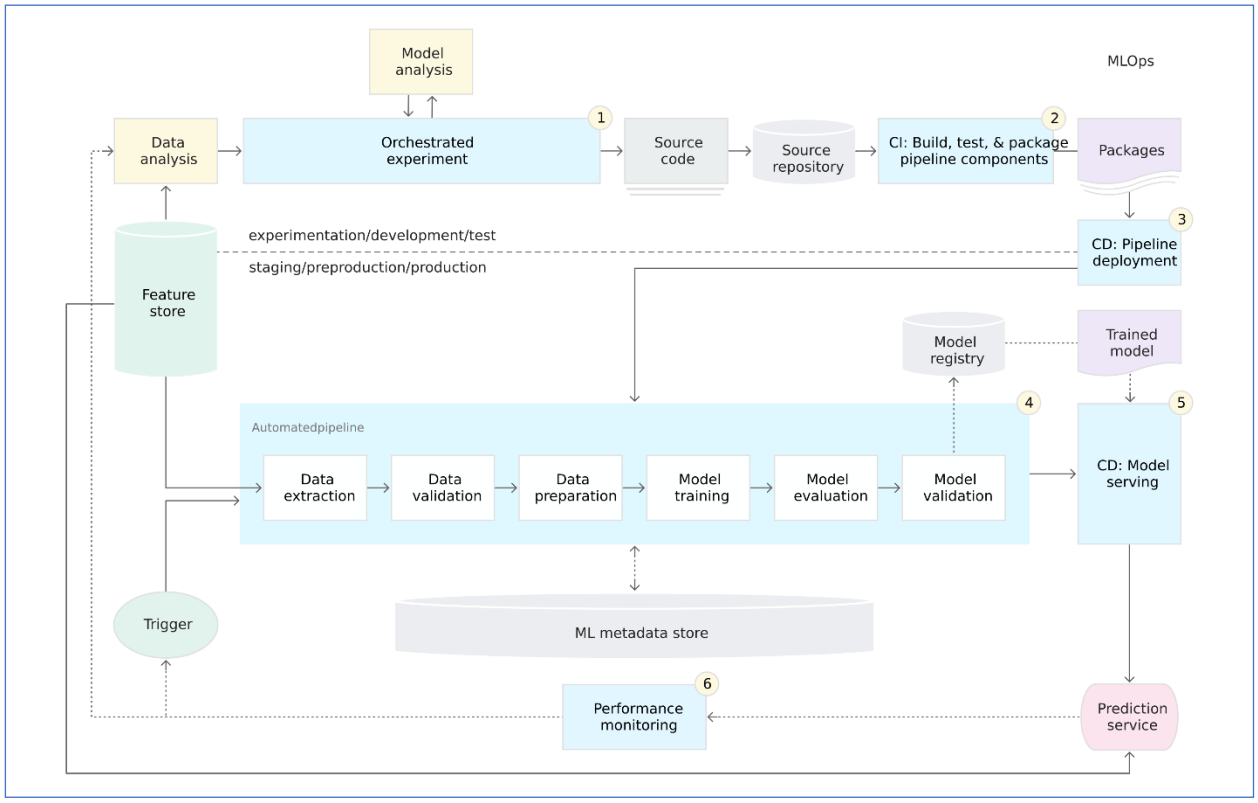


Figure 83. CI/CD and automated ML pipeline

This MLOps setup includes the following components:

- Source control
- Test and build services
- Deployment services
- Model registry
- Feature store
- ML metadata store
- ML pipeline orchestrator

Characteristics

The following diagram shows the stages of the ML CI/CD automation pipeline:

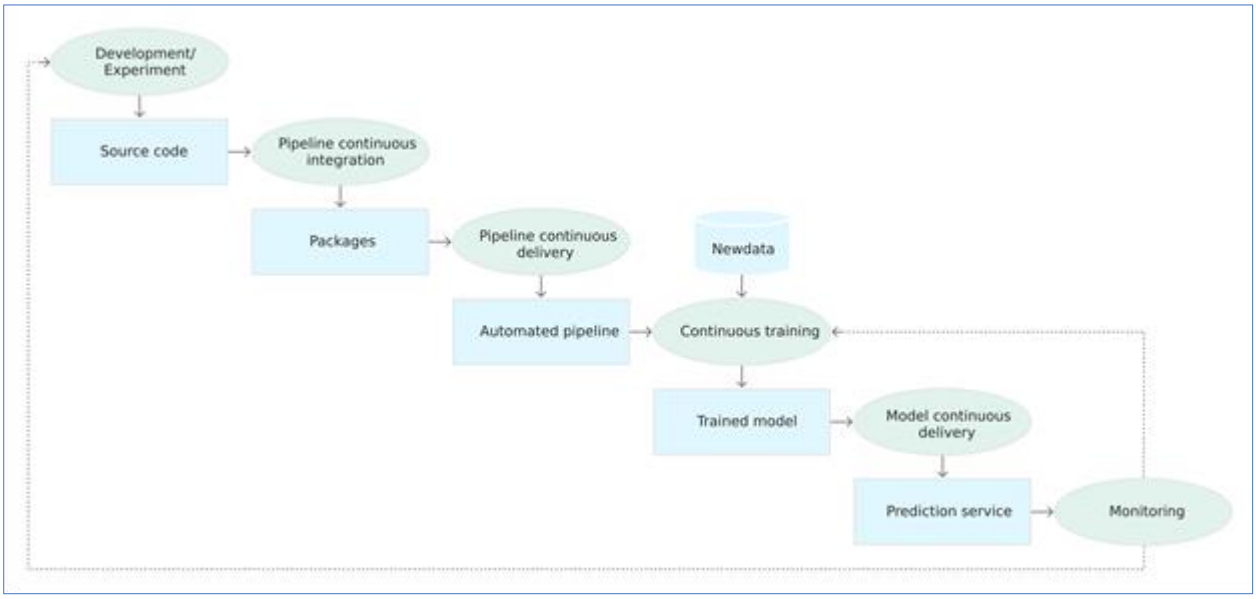


Figure 84. Stages of the CI/CD automated ML pipeline

The pipeline consists of the following stages:

- Development and experimentation: We iteratively try out new ML algorithms and new modeling where the experiment steps are orchestrated. The output of this stage is the source code of the ML pipeline steps that are then pushed to a source repository.
- Pipeline continuous integration: We build source code and run various tests. The outputs of this stage are pipeline components (packages, executables, and artifacts) to be deployed in a later stage.
- Pipeline continuous delivery: We deploy the artifacts produced by the CI stage to the target environment. The output of this stage is a deployed pipeline with the new implementation of the model.
- Automated triggering: The pipeline is automatically executed in production based on a schedule or in response to a trigger. The output of this stage is a trained model that is pushed to the model registry.
- Model continuous delivery: We serve the trained model as a prediction service for the predictions. The output of this stage is a deployed model prediction service.
- Monitoring: We collect statistics on the model performance based on live data. The output of this stage is a trigger to execute the pipeline or to execute a new experiment cycle.

The data analysis step is still a manual process for data scientists before the pipeline starts a new iteration of the experiment. The model analysis step is also a manual process.

Continuous Integration

In this setup, the pipeline and its components are built, tested, and packaged when new code is committed or pushed to the source code repository. Besides building packages, container images, and executables, the CI process can include the following tests:

- Unit testing our feature engineering logic.
- Unit testing the different methods implemented in model. For example, if we have a function that accepts a categorical data column and we encode the function as a one-hot feature.
- Testing that our model training converges (that is, the loss of our model goes down by iterations and overfits a few sample records).
- Testing that our model training doesn't produce NaN values due to dividing by zero or manipulating small or large values.
- Testing that each component in the pipeline produces the expected artifacts.
- Testing integration between pipeline components.

Continuous Delivery

In this level, system continuously delivers new pipeline implementations to the target environment that in turn delivers prediction services of the newly trained model. For rapid and reliable continuous delivery of pipelines and models, we should consider the following:

- Verifying the compatibility of the model with the target infrastructure before deploying the model. For example, we need to verify that the packages that are required by the model are installed in the serving environment, and that the memory, compute, and accelerator resources that are available.
- Testing the prediction service by calling the service API with the expected inputs, and making sure that we get the response that we expect. This test usually captures problems that might occur when we update the model version and it expects a different input.
- Testing prediction service performance, which involves load testing the service to capture metrics such as queries per seconds (QPS) and model latency.
- Validating the data either for retraining or batch prediction.
- Verifying that models meet the predictive performance targets before they are deployed.
- Automated deployment to a test environment, for example, a deployment that is triggered by pushing code to the development branch.

- Semi-automated deployment to a pre-production environment, for example, a deployment that is triggered by merging code to the main branch after reviewers approve the changes.
- Manual deployment to a production environment after several successful runs of the pipeline on the pre-production environment.

To summarize, implementing ML in a production environment doesn't only mean deploying the model as an API for prediction. Rather, it means deploying an ML pipeline that can automate the retraining and deployment of new models. Setting up a CI/CD system enables us to automatically test and deploy new pipeline implementations. This system lets us cope with rapid changes in our data and business environment. We don't have to immediately move all of your processes from one level to another. We can gradually implement these practices to help improve the automation of our ML system development and production.

4.2 Agile Development Methodology for Traditional Software Development

One of the prominent agile methodology in software development is Scrum based software development. To ensure that the product development is going based on timeline, without missing requirements and developer is not producing so many bugs – scrum gives a clean path on that. During overall development time Scrum framework would be followed as both development methodology and project management.

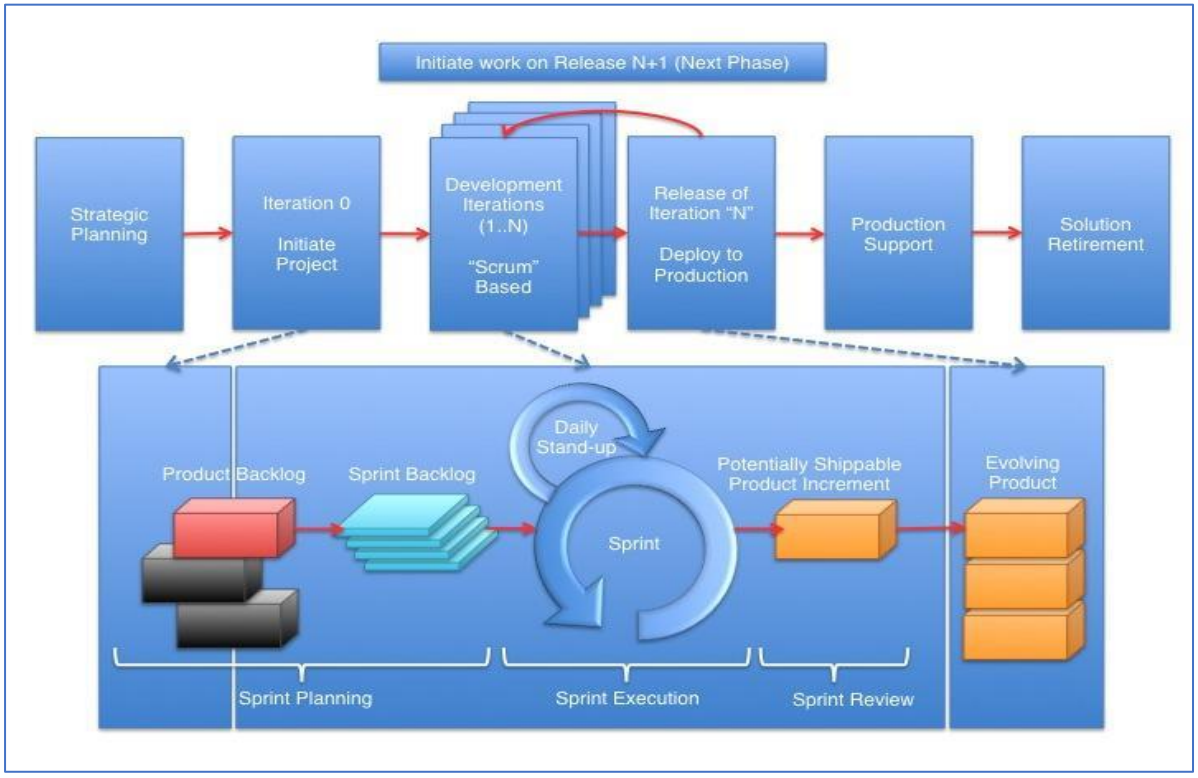


Figure 85. Methodology is a hybrid of SDLC activities and the Scrum development collaboration framework

4.2.1 Components of Scrum

This section in the Scrum overview will discuss common concepts in Scrum.

Product Owner: The product owner is the project's key stakeholder and represents users, customers, and others in the process. The product owner is often someone from product management or marketing, a key stakeholder, or a key user.

Scrum Master: The Scrum Master is responsible for making sure the team is as productive as possible. The Scrum Master does this by helping the team use the Scrum process, by removing impediments to progress, by protecting the team from outside, and so on.

Product Backlog: The product backlog is a prioritized features list containing every desired feature or change to the product. Note: The term "backlog" can get confusing because it's used for two different things. To clarify, the product backlog is a list of desired features for the product. The sprint backlog is a list of tasks to be completed in a sprint.

Sprint Planning Meeting: The product owner presents the top items on the product backlog to the team so that they can pick the item priority basis.

Daily Scrum: What has been achieved, what is in the plate and what the blocker – these three things are considered in the sprint planning meeting. All team members are required to attend the daily scrum.

Sprint Review Meeting: Typically, this takes the form of a demonstration of the new features, but in an informal way; for example, PowerPoint slides are not allowed. The meeting must not become a task in itself nor a distraction from the process.

Sprint Retrospective: Also, at the end of each sprint, the team conducts a sprint retrospective, which is a meeting during which the team (including its Scrum Master and Product Owner) reflect on how well Scrum is working for them and what changes they may wish to make for it to work even better.

4.2.2 Continuous Integration (CI) and Continuous Delivery (CD) Approach

Continuous integration (CI) and continuous delivery (CD) embody a culture, set of operating principles, and collection of practices that enable application development teams to deliver code changes more frequently and reliably. The implementation is also known as the CI/CD pipeline and is one of the best practices for DevOps teams to implement.

The goal of the continuous integration and continuous delivery (CI/CD) pipeline is to enable teams to release a constant flow of software updates into production to quicken release cycles, lower costs, and reduce the risks associated with development. Different set of tools and technologies will be used to implement continuous integration and continuous development.

Continuous integration (CI): CI is a software development practice in which small adjustments to the underlying code in an application are tested every time a team member makes changes. CI aims to speed up the release process by enabling teams to find and fix bugs earlier in the development cycle and encouraging stronger collaboration between developers– making it a crucial practice for agile teams. Historically, developers worked separately on parts of an application and would later integrate their code with the rest of the teams manually.

Depending on the when the next build occurred, it could take days, or even weeks, to see if the new code would break anything. This isolated process often led to developers duplicating their code creation efforts, developing with different coding strategies, and creating many hard to find and fix bugs. In a CI environment, developers merge their code changes in a shared repository several times a day so that it can be committed, tested, and validated continuously. Implementing CI speeds up the development process and ensures that bugs are caught earlier in the cycle.

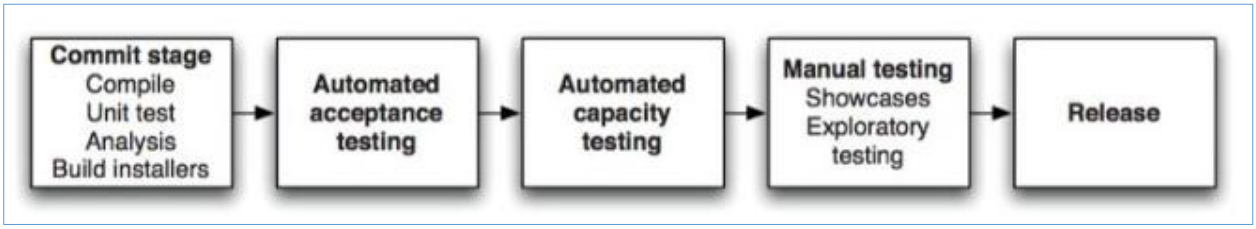


Figure 86. Continuous Integration Pipeline

Continuous delivery (CD): CD is the process of getting new builds into the hands of users as quickly as possible. It is the natural next step beyond CI and is an approach used to minimize the risks associated with releasing software and new features. Releasing software updates is notoriously painful and time-consuming. Continuous delivery reduces the risks and efforts associated with this process by ensuring that every change made to the underlying code of an application is releasable – meaning each update is smaller and can be delivered to users more frequently. By making releases less dramatic events that can be performed on-demand whenever new code is ready, teams can make their development process more efficient, less risky, and can get feedback from users quicker. If issues are found in production, they can be squashed quickly by simply rolling out the next update.

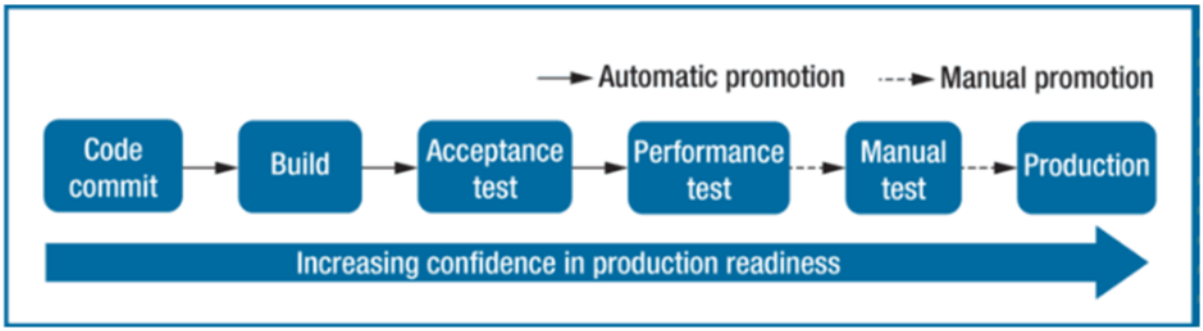


Figure 87. Continuous Delivery

4.3 Mobile Application Development Lifecycle

The chances of being able to deliver a successful app are extremely low unless we follow a documented app development process. This process will ensure that we don't waste valuable resources and time during the app development phase.

Figure 88. Mobile App Development Life Cycle

4.3.1 Planning Expression/Conceptualization

The first step is the most important as it is the foundational step for app development requirements. It lays the groundwork for the next step and thus a thorough research and ideal brainstorming is recommended before moving onto the next step. This section also covers Prototyping which also known as building mockups.

- Aware of target audience: Decide right user experience. App shouldn't go overboard in supporting all devices along with all platforms.
- Building App Prototype: Before jumping on mobile application development, one must make a prototype of an app that gives a well-defined preview of the application final view. In this process it'll be visible what app would feel when it is touched, the interface, and the touch features everything through which will be able to make optimizations in designs.

- Finalizing application mockups: App should follow mobile specific UI guidelines provided by Google. One tool for such wire framing is MockFlow.

- Decide on data exchange is crucial: Decide data exchange protocols between data server and mobile application in advance like REST or SOAP or JSON or XML.

- Going for multi-layer instead of single layer:
    - User Interaction – Native UI controls, HTML etc.
    - Business Logic – Define logics in java classes.
    - Data access layer – Database related stuff.

4.3.2 Application Development Phase

Conversion of concrete form into executable code. This is the phase where actually coding will be done to build the app from ground.

- **Act Smart:** Developer should keep in mind all the limitation of mobile platforms be it device related or network ones. Best is to follow mentioned guidelines for e.g., guidelines are different while designing UI for tablets and mobile devices.

- **Develop UI of the App:** Designing of apps comes a long way before development. UI should be beautiful and stunning with user-friendly app interface, decide the flow of app, touch and feel of it etc. Creating a user environment for application is a long process but a very important one. After that, the visual directions and blueprints will be apparent which shapes the final product.

- **Design Patterns:** MVC and Template patterns can be used to save development efforts and standardize solutions in more proven and manageable way.

- **Maintain code's quality:** Jupiter can be used for static analysis. Also, Versioning is very very important. It'll be pretty much impossible to maintain code throughout the team without having a proper Version Control. GIT or any other source code version tool will surely use to achieve this.

- **Developer Testing:** There are various developer testing methodology and tools are available. We're preferring JUnit because of its vast popularity and acceptability.

4.3.3 Testing

On completing mobile application development, another most important phase starts and that is testing an application. This phase helps check app's environment, functionality, features, debugging.

- **Automation:** Automate the testing – UI testing using FoneMonkey or MonkeyTalk and Data validation testing using TestLodge or any other tool feasible for all of these.
- **Device debugging**
- **Capturing all crash reports**
- **Aggressive Testing:** Test on different devices, network types and available resources so that performance of app can be tracked.

5. Client - Server Architecture

5.1 Client-server Architectures

In the client-server architectural model – the resources are given in the server so that client can access. More specifically this is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. Sometimes client-server architectures are commonly organized into layers referred to as “tiers”.

5.2 Client

Client is sort of a gateway or an interface which let’s use the product. It is a layer between a user and some complex implementation which runs the product, Client takes care of providing the user with abstracted layer of interface through which user ends up using the product.

5.3 Server

Server is nothing but a compute compatible machine which runs the product and gets requests from clients, Which in turn is receipted and decoded to understand what is it client wants. Servers are responsible for running the product and perform the tasks requested by the client.

5.4 Docker Server

Docker as we all know by does all things containers. It provides a container runtime, It is manages container storage, networks, image builds and many other services. No doubt it requires compute power to perform all these tasks. Docker server is nothing but a compute instance (VM, Personal Computer etc.) which is responsible for running Docker daemon process which in turn manages all the services provided by Docker.

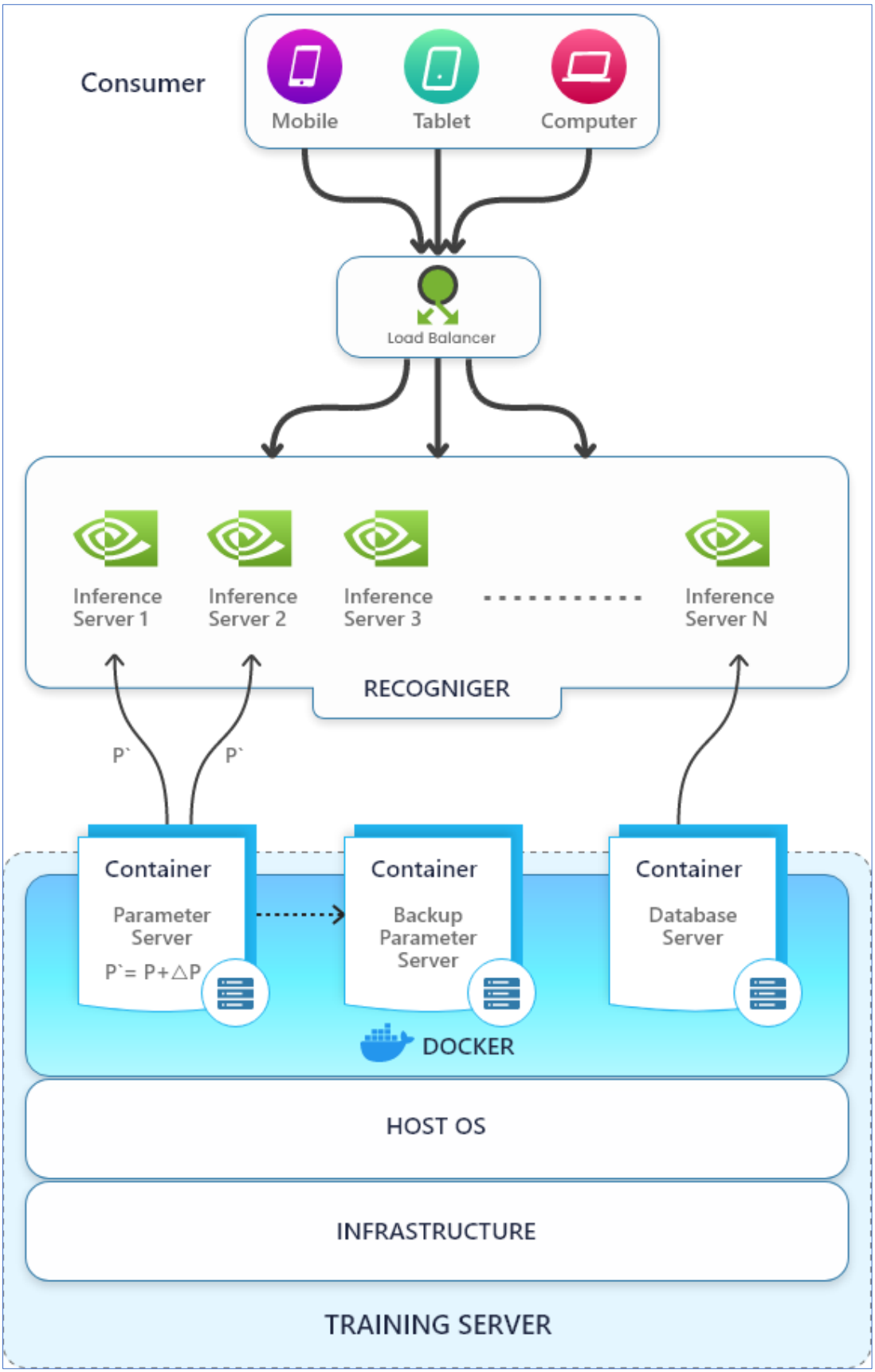


Figure 89. Client Server Architecture

6. Milestone, Duration & Methodology

6.1 Work Plan & Project Duration

6.1.1 Activity Plan & Timeline

Activity Plan: Our comprehensive plan of development, training and testing for SLR, T2SP, Braille converter, web application and mobile apps has been split into multiple activity with multiple version. Here below is our activity and timeline for development, training, testing and finally the implementations. The activity which is exclusion of development, training and testing of Engine/DL model, categorized as "Client & API". In the visual timeline "0-18" denoted the month counting.

- Reporting
    - Inception Report (Client & API)
- Sign Language Recognition (SLR)
    - Engine
        - Gold standard data (Client & API)
        - Crowdsource (Client & API)
    - Web application (Client & API)
        - API expose
    - Android & iOS application (Client & API)

- Text to Sign Puppet (T2SP)
    - Engine
    - Web application (Client & API)
    - Mobile application (Client & API)

- Braille Converter
    - Desktop application
    - Office Add-ons
    - Book printing

Below structure is the breakdown of activity of each module for development, training and testing –

Sign Language Recognition (SLR)

- Basic SLR v-1.0
    - Hypothesis Testing
    - Annotation format fixing
    - Model fixing
    - Data Labeling

- SLR v-1.1
    - Post processing the labeled data
    - Creating pipeline for new data
    - Analysis new data
    - Building supporting NLP Corpus
    - Corpus Validation
    - Gloss2test training with corpus
    - Gloss2text validation

- SLR v-1.2
    - Word2vec model training
    - Word tokenizer
    - Adding new data with pipeline
    - Optical flow model training
    - PostNet model training
    - Object detector model training

- SLR v-1.3
    - All model analysis with WER rate
    - Tuning model with hyper-parameters
    - Autotuning or augmenting the data process

- Validating the model

- SLR v-2.0
  - New data edition and training all models
  - Performance testing of data
  - Tuning the model WER decrease rate
  - Dockerizing the model
  - Model deployment
- SLR v-3.0
  - Fine Tuning for performance improvement after getting UAT feedback

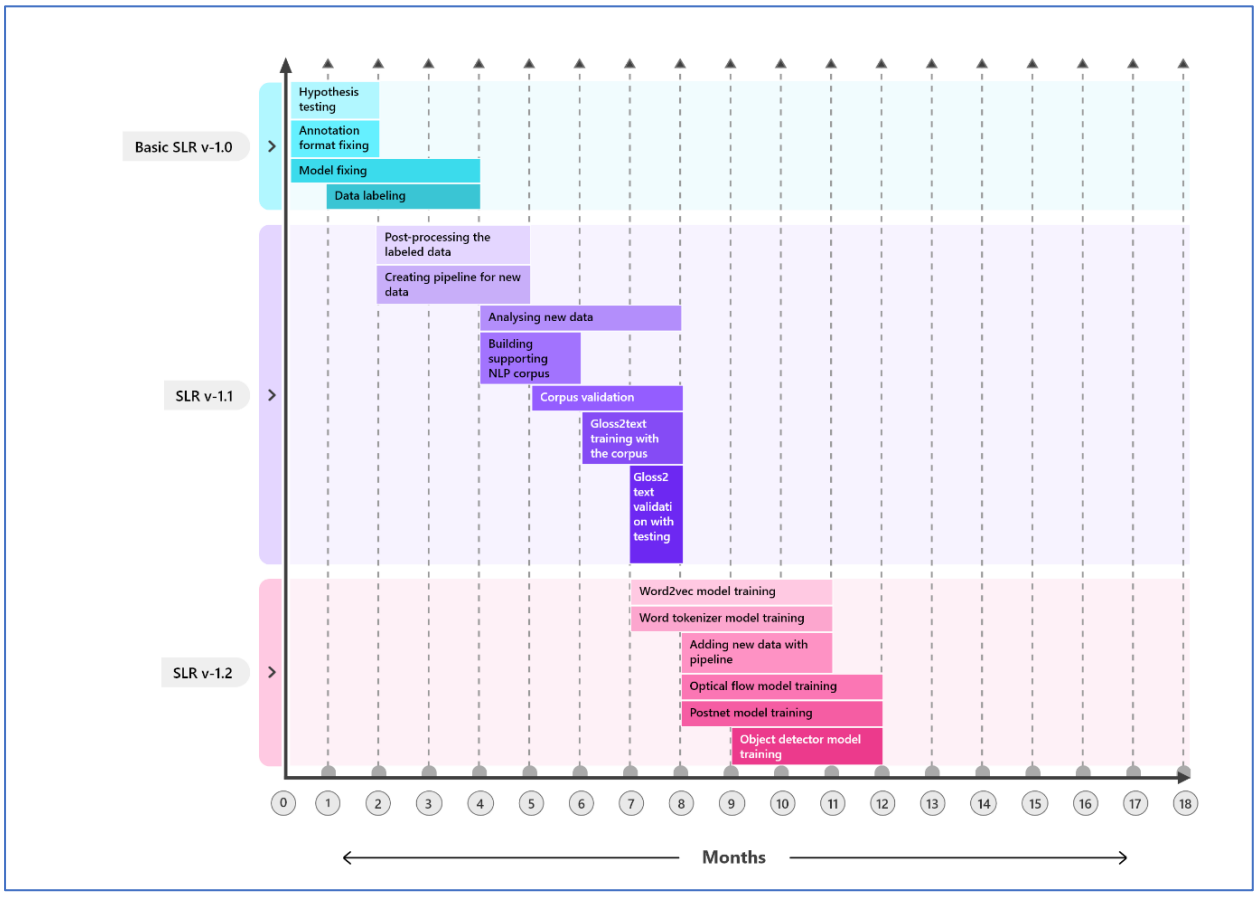


Figure 90. SLR timeline-1

Basic SLR v-1.0 starts with phases, "hypothesis testing", "annotation format fixing" & "model fixing" from the first month and phase "object detector model training" of SLR v-1.2 finishes at the end of eleventh month. (Continues...). Number 0 to 18 shown in the timeline is counting for months with project duration.

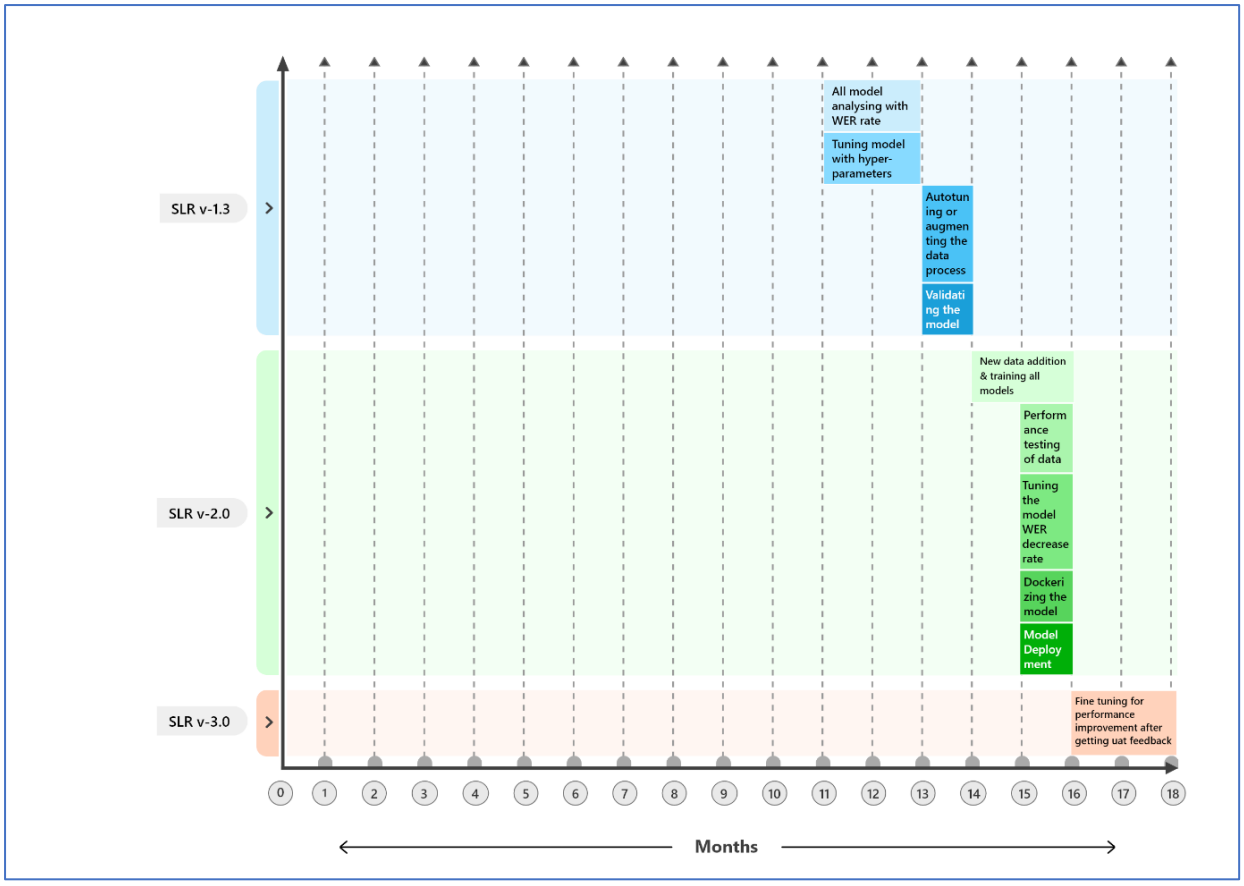


Figure 91. SLR timeline-2

SLR v-1.3 starts with phases, "All models analyzing with WER rate" & "Tuning model with hyper-parameters" from the eleventh month and phase "fine tuning for performance after getting UAT feedback" of SLR v-3.0 finishes at the end of seventeenth month. Number 0 to 18 shown in the timeline is counting for months with project duration.

Text to Sign Puppet (T2SP)

- Basic T2SP v-1.0
    - Hypothesis testing
    - Annotation format fixing
    - Labeling data with manual
    - Labeling data with auto-process
    - Validating the annotation data
- T2SP v-1.1
    - GAN model training
    - Bangla corpus building

  - Validating Bangla corpus
  - Training NLP models
  - Validating NLP models
  - Validating GAN model
- T2SP v-1.2
  - Tuning GAN model
  - Tuning NLP model
  - New data addition
  - Hyper-parameter tuning
  - Training all models
  - Analysis WER rate
- T2SP v-2.0
  - Re-training the models to optimize the WER
  - Validating and testing
  - Dockerizing the model
  - Deploying and testing the model
- T2SP v-3.0
  - Fine tuning for performance improvement after getting UAT feedback.

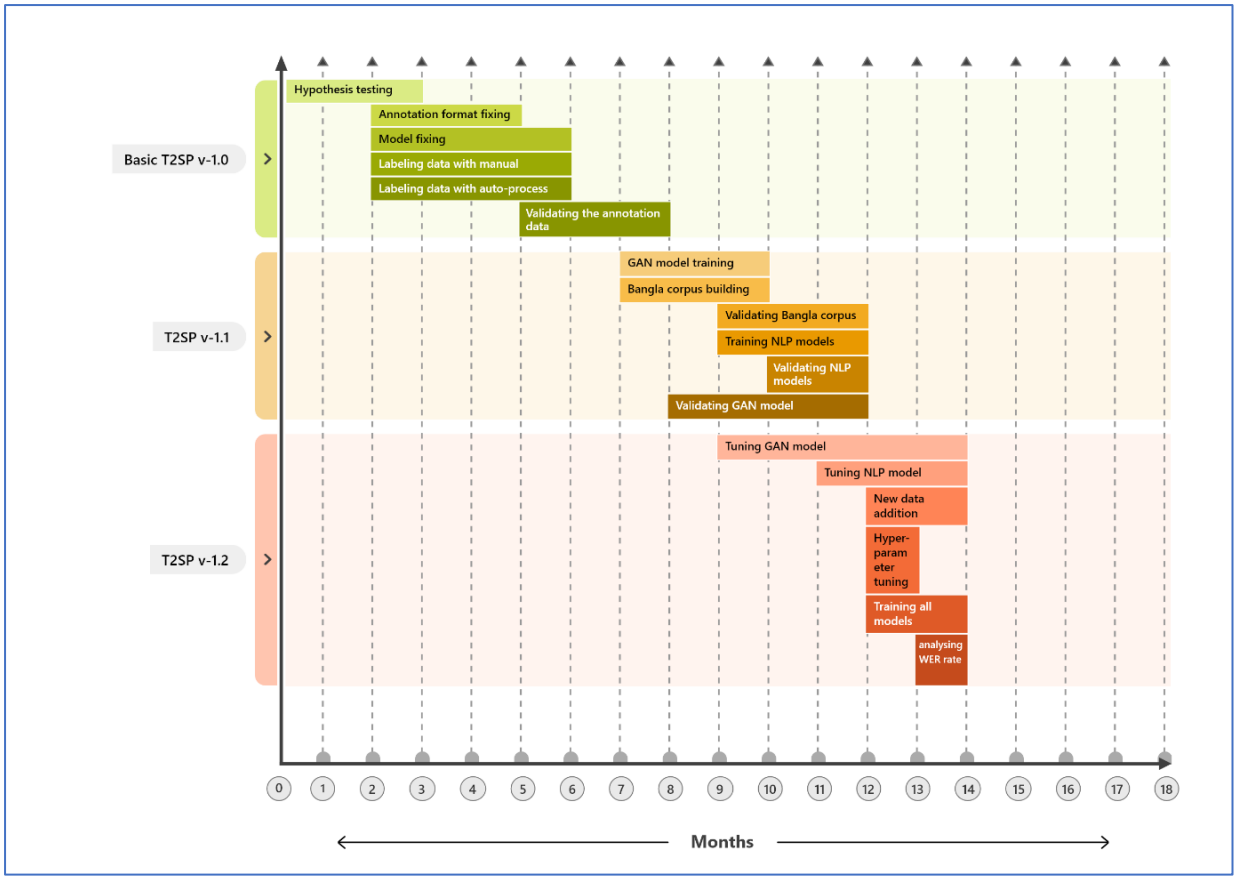


Figure 92. T2SP timeline-1

Basic T2SP v-1.0 starts with phases, "Hypothesis testing" from the first month and phase "Analyzing WER rate" of T2SP v-1.2 finishes at the end of thirteenth month. (Continues...). Number 0 to 18 shown in the timeline is counting for months with project duration.

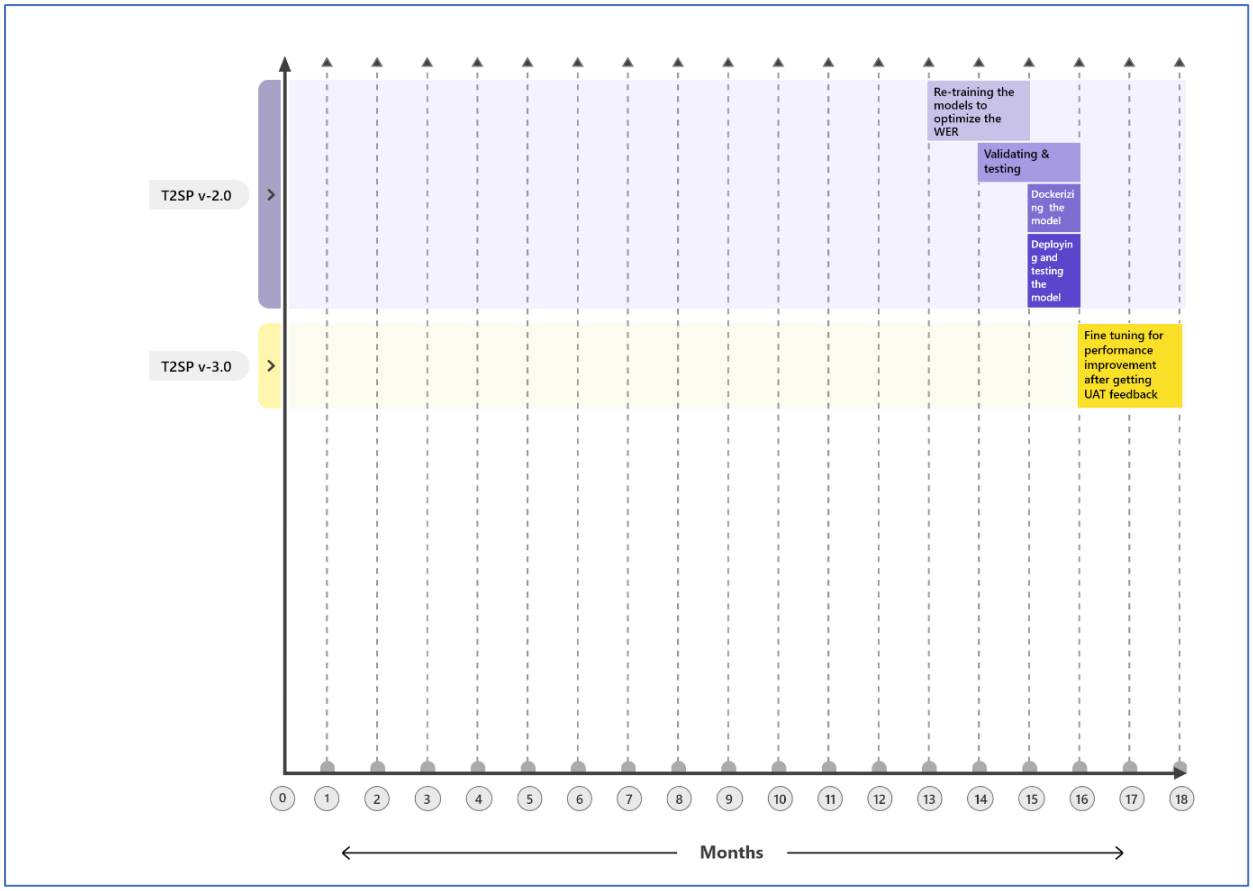


Figure 93. T2SP timeline-2

Basic T2SP v-2.0 starts with phases, "Re-training the models to optimize the WER" from the thirteenth month and phase "fine tuning for performance after getting UAT feedback" of T2SP v-3.0 finishes at the end of seventeenth month. Number 0 to 18 shown in the timeline is counting for months with project duration.

Bangla to Braille Converter (B2BC)

- B2BC v-1.0
  - Bangla Unicode to braille converter development
- B2BC v-1.1
  - Development and integration of editor with converter
- B2BC Office Extension
  - B2BC office extension development
- Braille book printing
  - Printing 5 books braille using converter (50 copies of each book)

- Printing 20 books braille using converter (50 copies of each book)
- Printing 75 books braille using converter (50 copies of each book)

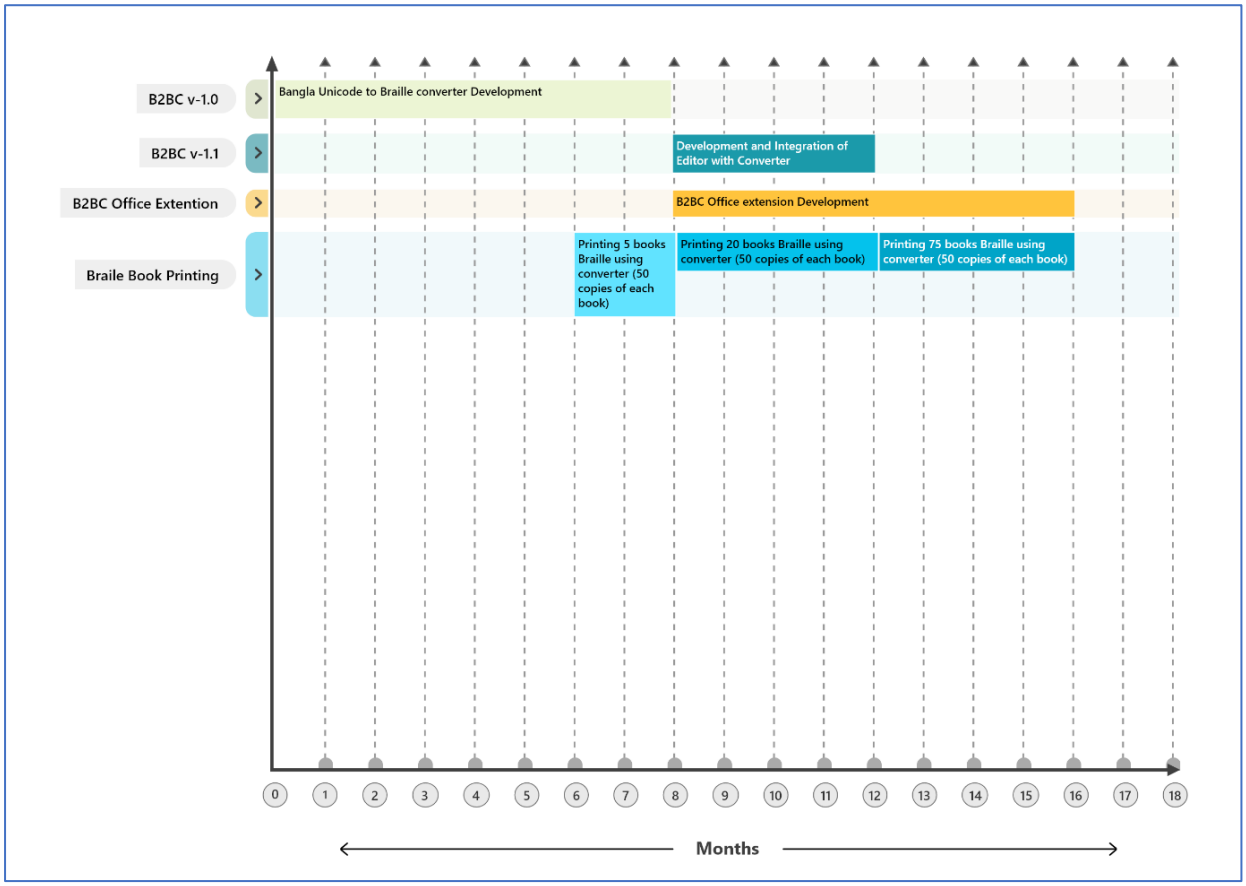


Figure 94. B2BC timeline

B2BC v-1.0 starts with phases, "Bangla Unicode to Braille converter development" from the first month and phase "Printing 75 books using braille converter" of braille book printing finishes at the end of fifteenth month. Number 0 to 18 shown in the timeline is counting for months with project duration.

Client & API

- Reporting
  - Inception report
- Gold standard data
  - 25k sentences generation
  - Video recording for 25k sentences
  - Video labeling
  - (Ultimately 500k sentence generation)
  - (Ultimately video recording for 500k sentences)

  - (All video labeling)
- SLR crowdsource v-1.0
  - UX design
  - Interface development
  - Dataset collection & processing interface
- SLR crowdsource v-1.1
  - Crowdsourcing and validation module
- Basic web SLC v-1.0
  - UX mockup design
  - Interface development
  - Streaming module
  - Broadcasting features on pair communication
- Web SLC v-1.1
  - Basic STT and TTS integration
- Web SLC v-2.0
  - API integration with T2SP dockerized model
  - API integration for recognition engine
- Web SLC v-3.0
  - Fine tuning for performance improvement
- Basic mobile app (Android & iOS) v-1.0
  - UX mockup design
  - Interface development
  - Chat feature development
  - Mobile app's pivotal feature development
  - Segregate the model
  - Test the model
- Mobile app (Android & iOS) v-1.1
  - Performance analysis
  - Convert the model

  - M-GPU release or lite training
  - Validating with low devices
- Mobile app (Android & iOS) v-2.0
  - SDK integration for realtime video
  - Model testing
  - Creating a model for multi-cluster
  - Testing & deploying the model
- Exposing API
  - API for developer of the recognition engine
  - API for corpus/Data transfer
  - API for third party service
- Web-1.0
  - UX for landing portal
  - Development backend
- Web-1.1
  - Integrate SLR v-1.0
  - Basic T2SP v-1.0
- Web-1.2
  - Integrate SLR v-1.1
  - Integrate basic mobile app v-1.0
  - Integrate web SLC v-1.0
- Web-1.3
  - Integrate web SLC v-1.1
  - Integrate T2SP v-1.2
  - Integrate basic mobile app v-1.2
- Web-2
  - Integrate management interface
  - Integrate web SLC v-2.0
  - Integrate T2SP v-2.0

- Integrate basic mobile app v-2.0
- API integration with SLR engine
- API integration with corpus/data transfer
- API integration with third party

- Web-3
  - Fine tuning for performance improvement after getting UAT feedback

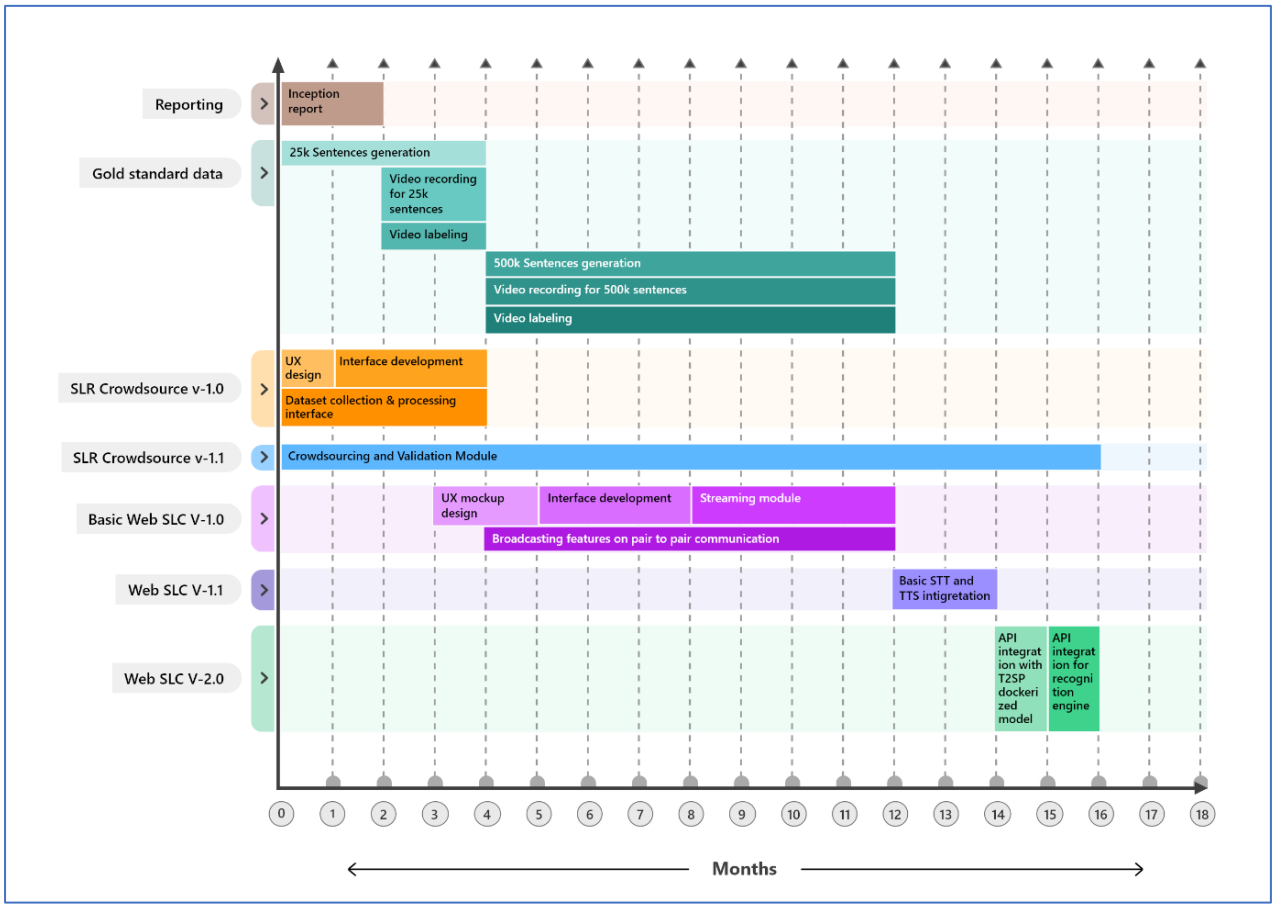


Figure 95. Client & API timeline-1

Client & API first segment starts from the first month with preparation and completion of Inception Report (in two months), Gold Standard Data (in ten months), SLR Crowdsource v-1.0 (in four moths), SLR Crowdsource v-1.1 (in sixteen months), Basic Web SLC v-1.0(in ninth months), Web SLC v-1.1(in two months) & ends with Web SLC v-2.0 (in two months).

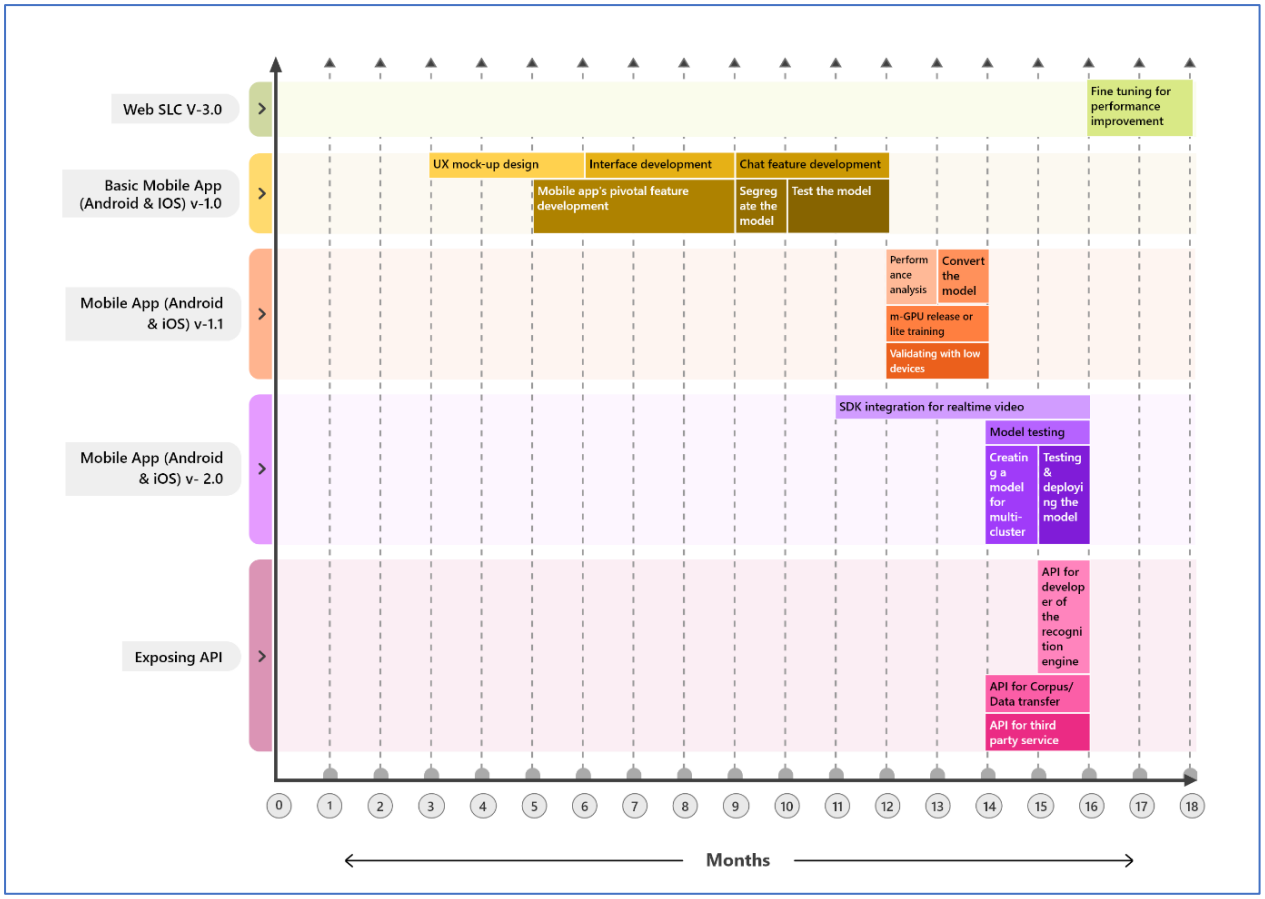


Figure 96. Client & API timeline-2

Client & API second segment starts from the third month with preparation and completion of Basic Android & iOS mobile app v-1.0(in nine months), Android & iOS mobile app v-1.1(in two months), Android & iOS mobile app v-2.0(in five months), Exposing API (in two months) & ends with Web SLC v-3.0 (in two months)

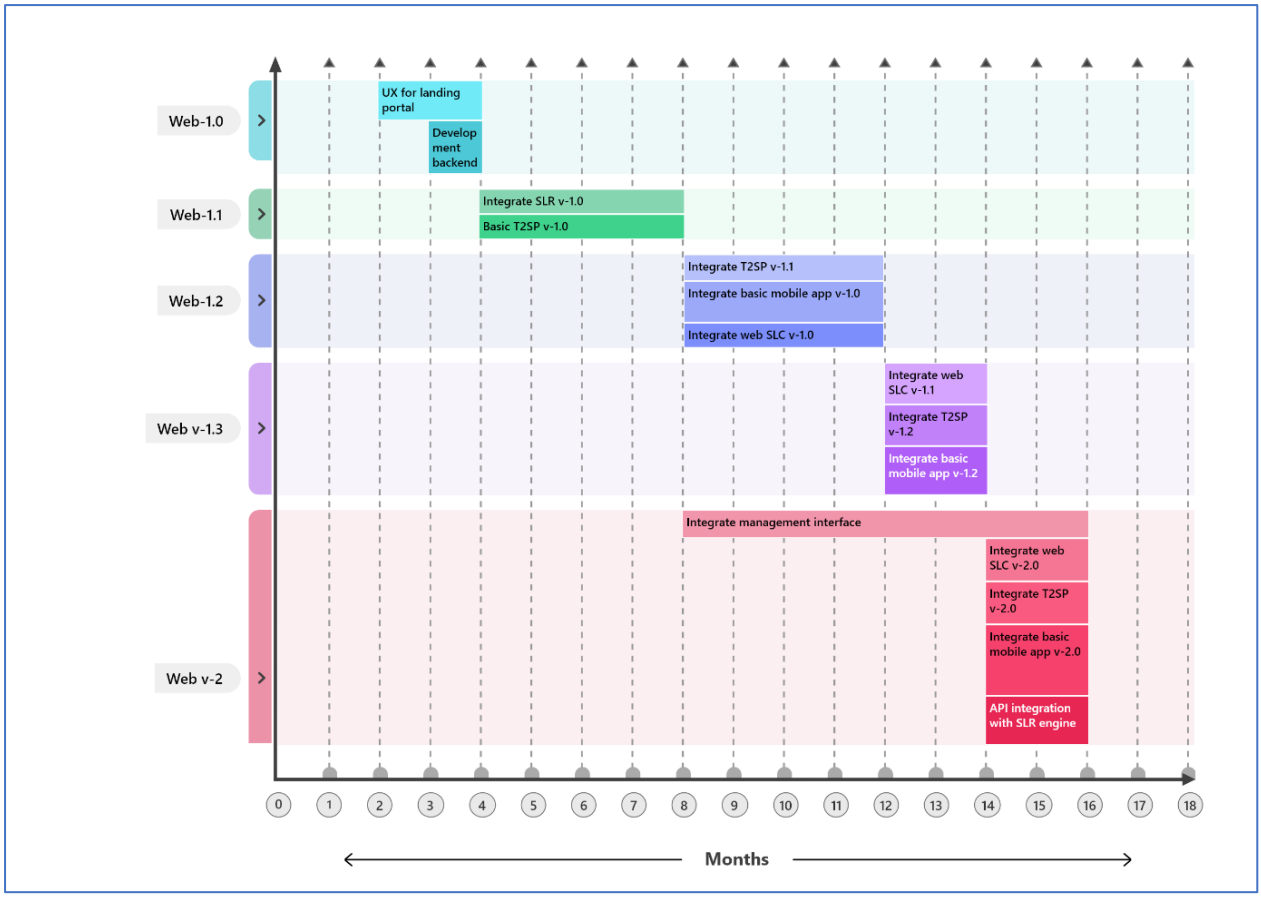


Figure 97. Client & API timeline-3

Client & API third segment starts from the second month with preparation and completion of Web v-1.0(in two months), Web v-1.1(in four months), Web v-1.2(in four months), Web v-1.3(in two months) & ends with Web v-2.0 partially.

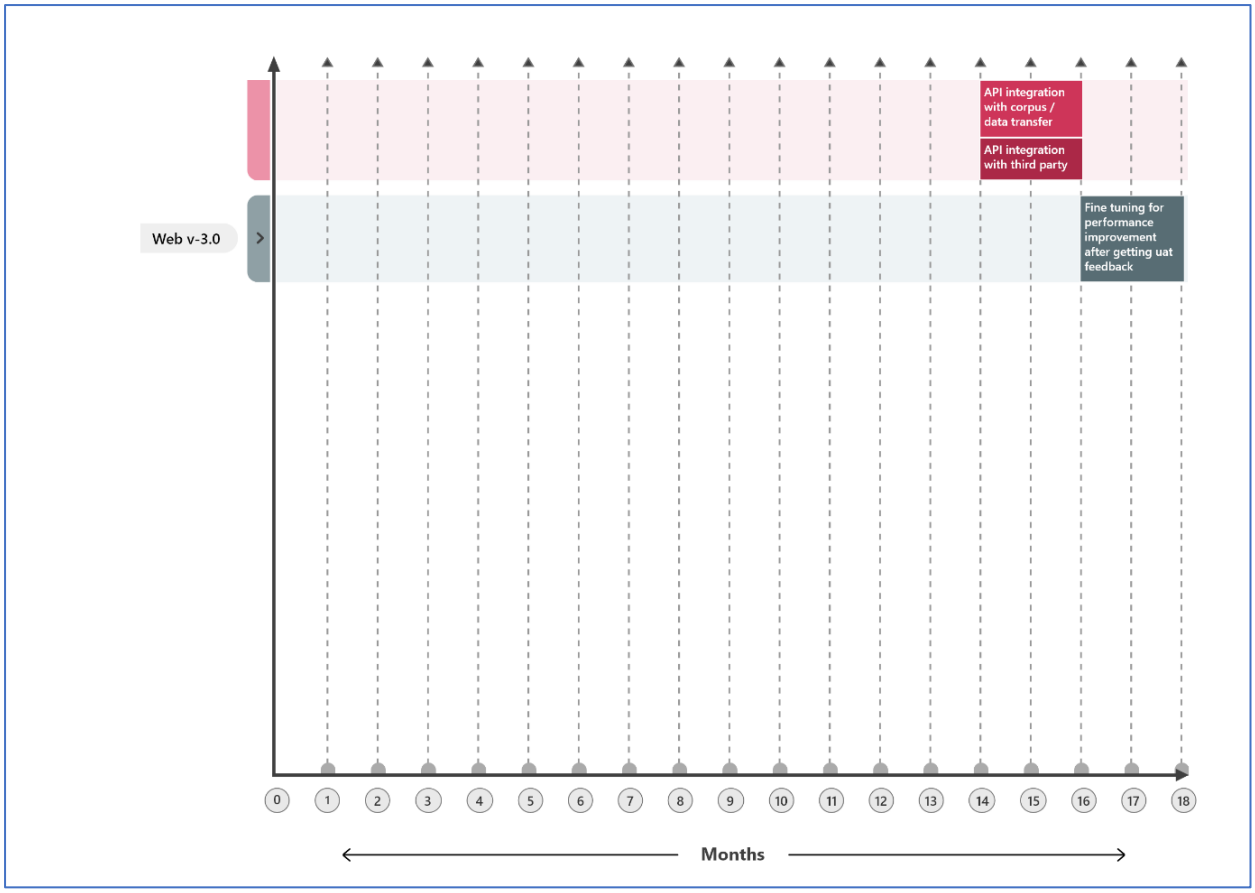


Figure 98. Client & API timeline-4

Client & API fourth & final segment starts from the fourteenth month with preparation and completion of leftover Web v2.0 (in total ten months) & ends with Web v-3.0(in two months).

6.2 Test Plan & Test Schedules

6.2.1 Test Plan Development & Execution

We will take an approach of test-driven development for SD-17. A separate quality assurance team from Genweb2 will work independently parallel to the development team for this purpose. Project team and the quality assurance team will work together to generate the test cases. However, the quality assurance team will be mainly responsible for performing the tests and submit reports to the project team. Test cases will be written for every part of the system right from the beginning of development. The developers will keep developing the system and try to pass these primary test cases to confirm the appropriate process flow and expected functionality of the system. The test-cases will be reviewed by the team of professional when listed. After development is complete, the full application will be tested comprehensively. The test outcomes will be monitored, and consultants' own testing will go on in parallel. Testing plan will introduce variety in user group. The user group will include the following categories of people –

- Genweb2 Testing Team
    - ML Engineer
    - Application and API Tester
    - QA Engineer
    - Software Engineer
- Linguistic Experts
- Researchers
- Beta Tester (End User like deaf people)

Test Plan Development

We will discuss different types of testing to be performed in this project. Following testing category will be performed throughout the development life cycle of the project

- Model Testing
- Application Testing

6.2.1.1 SLR

6.2.1.1.1 Model testing

Hand and facial gesture are the core term of Sign Language. Vision-based Sign Language recognition is an area of active research in computer vision and machine learning where deep learning has a direct connection with this sign language recognition. Application quality in Machine learning and Deep learning systems are different. In this, accuracy, robustness, learning efficiency and adaptation and performance of the system checked. The elements which considered for testing in the case of ML is Models.

Model Testing steps

- Based on historical data (previously verified defects) creating a training data set.
- Train predictive model using the classification algorithm that meets defined quality criteria.
- The model has exposed as an online REST service as we quickly call it and make prediction requests for new incoming defects.
- Defect records that store in a code repository automatically updated with the scoring result: prediction and probability of prediction.
- Defects are sorted by prediction probability, so QA engineer could start testing bugs with the highest probability of fix (incorrect).
- If new bugs successfully verified, also label value is known (correctly fixed or not), such records are marked as new training data. Then it stored in the feedback data store.

Both testing practices and results have changed to accommodate applications that don't behave the same as traditional software. Traditional testing techniques based on fixed inputs. We knew that given inputs **x** and **y**, the output will be **z** and this will be constant until the application changes. It is not true in machine learning systems. The output does not fix. It will change over time. It is similar to the model built on Machine Learning system evolving as more data fed. It forces us to think differently and adapt test strategies that are very different from traditional testing techniques. Critical activities that will be essential to test machine learning systems –

Developing training data sets - This refers to a data set of examples used for training the model. In this data set, you have the input data with the expected output. This data usually prepares by collecting data in a semi-automated way. The goal of such a continuous learning system is to ensure the highest possible quality of the exposed model. The feedback data store is used to evaluate the quality of the server model, automatically retrain the deployed model, and finally to re-deploy the new model version.
Developing test data sets - It is a subset or part of the training dataset that is built to test all the possible combinations, also estimates how well the model trains. Based on the test data set results, the model fine-tuned.

Model Testing Tools
For manual testing of a Machine learning, the tools which can be used to develop a machine learning model can also be used for testing the model. The tools are –

- WEKA
- PyCharm
- Spyder

6.2.1.1.2 Application testing

Usually, application testing includes:

UI/UX tests: Basically for any application, whether it's a web or mobile application, UI testing is mandatory. It's the test where the interface padding with screen resolution, section breaking and other interface related problems are checked.

Unit tests: The application build program is broken down into blocks, and each element (unit) is tested separately.

Integration tests: This type of testing observes how multiple components of the program work together.

Regression tests: This cover already tested application to see if it doesn't suddenly break.

Functional tests: The application needs to get well communicate with hardware like camera and focus on vision. Takes the vision and run simulation upon that with trained algorithm. Returns expected results as in Bangla Text.

Performance tests: The application should properly handle all the required functionality in proper/specific time and process, without broken though there is massive pressure of traffic.

Security test: The security testing will involve an active analysis of the SLR application for any weaknesses, technical flaws, or vulnerabilities. The primary purpose will be to identify the vulnerabilities

6.2.1.2 T2SP

6.2.1.2.1 Model testing

Mapping of Bengali text/Sentences and Avatar database

Prepare the model and test dataset such a way so that we can justify test results touches our set approximation. A mapping database of Bengali sentence/gloss and avatar is there. Test dataset and model dataset considered to be ready for getting the closer matching of the avatar movements. The more test dataset we will have, the more accurate result we will find. Like before, for the training dataset of SLR we need to test avatar test dataset also.

Evaluation

For the evaluation of avatar movement, we will use gesture generator to generate gesture animations with testing data, and to evaluate the quality of generated animations, we compare them with two gesture animations:

- The original motion capture data of the testing data.
- The motion capture data of the test case actor with respect to other text/sentences.
- In terms of accuracy of avatar, techniques below will be followed:

    - Tracking avatar head movements - The static head markers have to be tracked to obtain information about head movements. In addition, the several different facial surface markers must be tracked for the short parallax phase of the video clip. Subsequently, from the parallax displacement of the tracked markers for

the different frames of the video clip, 3D coordinates for each tracked marker need to be measured.

- Tracking the Avatar facial surface - The individual 3D surface of the participant's (while recording video) face is built on the basis of the 3D coordinates of the several different surface markers. First, the facial surface is constructed by connecting four markers at a time to form rectangles - the connection lines between the facial surface markers. Subsequently, this rough approximation of the facial surface by the rectangles is interpolated and smoothed to closely fit the participant's real facial surface. Afterward, the tip of the nose in the facial surface is centered at the origin of Blenders 3D coordinate system. This way all the recorded data path is compared with the generated avatar to find the close approximation.
- Tracking Avatar emotion markers - The facial emotion markers must be tracked for the emotionally relevant episodes from the video clips. The movement of these markers is also exported into respective tool like Blender's 3D space, namely as a projection from the moving camera onto the static facial surface. The movement of these markers on the facial surface represents the movement of the markers painted on the participants 'skin. A python script uses Blender's application programming interface (API) to access the 3D coordinates of the emotion markers per frame, exports the coordinates, and saves them in a comma-separated values file (CSV) for each participant for further data processing. A comparison can be made with the generated avatar and the recorded clip when 3D model was made.
- Scaling and standardization procedures - Two individual scaling procedures might be followed for the generated modeling data to be rescaled into meaningful measurement units. The process of taking the data from the tracks of the video clip and transforming them into Blender's 3D space (A modeling tool) cannot be controlled and is therefore relatively arbitrary. Therefore, accurate measurement of a real-world object of known size is needed for the rescaling. Different techniques will be followed for rescaling.

- Accuracy of the measurement procedure - To estimate the accuracy of the tracking procedure, need to calculate the following parameters:
  - Accuracy of measuring head movements and estimating the facial surface.
  - Accuracy of measurement of the scaling parameters.
  - Accuracy of the model building and rescaling.

6.2.1.3 B2BC

6.2.1.3.1 B2BC application Testing

UI/UX tests: For Bangla to braille application, UI testing is not mandatory as it's a desktop application. All the buttons and entry field including label panel padding is going to be checked here.

Unit tests: The application build program is broken down into blocks, and each element (unit) is tested separately.

Integration tests: This type of testing observes how multiple components of the program work together.

Regression tests: This cover already tested application to see if it doesn't suddenly break.

Functional tests: The application needs to get well communicate with hardware like printer and focus printer connection port.

Performance tests: The application should properly handle all the required functionality in proper/specific time and process, without broken though there is massive pressure of printing instruction.

6.2.1.3.2 B2BC Add-ons/Extension Testing

Integration tests: This type of testing observes how multiple components of the program work together.

Functional tests: The application needs to get well communicate with primary software integration like Microsoft Word.

Performance tests: The application should properly handle all the required functionality in proper/specific time and process, without broken though there is connection of externally add-ons.

6.2.1.4 API Testing

API testing is a type of software testing that involves testing Application Programming Interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security. Since APIs lack a GUI, API testing is performed at the message layer. API testing is now considered critical for automating testing because APIs now serve as the primary interface to application logic.

API testing is used to determine whether APIs return the correct response (in the expected format) for a broad range of feasible requests, react properly to edge cases such as failures and unexpected/extreme inputs, deliver responses in an acceptable amount of time, and respond securely to potential security attacks. Service virtualization is used in conjunction with API testing to isolate the services under test as well as expand test environment access by simulating APIs/services that are not accessible for testing.

Types of API tests

Functional testing ensures the API performs exactly as it is supposed to. This test analyzes specific functions within the codebase to guarantee that the API functions within its expected parameters and can handle errors when the results are outside the designated parameters.

Load testing is used to see how many calls an API can handle. This test is often performed after a specific unit, or the entire codebase, has been completed to determine whether the theoretical solution can also work as a practical solution when acting under a given load.

Reliability testing ensures the API can produce consistent results and the connection between platforms is constant.

Security testing is often grouped with penetration testing and fuzz testing in the greater security auditing process. Security testing incorporates aspects of both penetration and fuzz testing, but also attempts to validate the encryption methods the API uses as well as the access control design. Security testing includes the validation of authorization checks for resource access and user rights management.

Penetration testing builds upon security testing. In this test, the API is attacked by a person with limited knowledge of the API. This enables testers to analyze the attack vector from an outside perspective. The attacks used in penetration testing can be limited to specific elements of the API or they can target the API in its entirety.

Fuzz testing forcibly inputs huge amounts of random data -- also called noise or fuzz -- into the system, attempting to create negative behavior, such as a forced crash or overflow.

API testing tools

When performing an API test, developers can either write their own framework or choose from a variety of ready-to-use API testing tools. Designing an API test framework enables developers to customize the test; they are not limited to the capabilities of a specific tool and its plugins. Testers can add whichever library they consider appropriate for their chosen coding platform, build unique and convenient reporting standards, and incorporate complicated logic into the tests. However, testers need sophisticated coding skills if they choose to design their own framework.

Conversely, API testing tools provide user-friendly interfaces with minimal coding requirements that enable less-experienced developers to feasibly deploy the tests. Unfortunately, the tools are often designed to analyze general API issues and problems more specific to the tester's API can go unnoticed.

A large variety of API testing tools is available, ranging from paid subscription tools to open-source offerings. Some specific examples of API testing tools include:

SoapUI: The tool focuses on testing API functionality in SOAP and REST APIs and web services.

Apache JMeter: An open-source tool for load and functional API testing.

Apigee: A cloud API testing tool from Google that focuses on API performance testing.

REST Assured: An open source, Java-specific language that facilitates and eases the testing of REST APIs.

Swagger UI: An open-source tool that creates a webpage that documents APIs used.

Postman: A Google chrome app used for verifying and automating API testing.

Katalon: An open-source application that helps with UI automated testing.

Key benefits of API testing

- Less time consuming than GUI functional testing
- Can be done early in the development cycle without waiting for GUI development to complete
- Can help to stabilize business logic and help in efficient GUI integration
- Lightweight XML or JSON data transfer modes
- Can help in issue localization and isolation during GUI integration
- Automation can be done in any programming language
- Helps in regression testing

Handful API Testing for EBLICT Project

User Category	Test Process	Test Result
Project Owner	API takes the generic request	Returns the requested feasible result
3rd party	API takes the user request, Validating of user registration.	If user registered, then the API is taking user's specific request and returns only the registered service result, otherwise ask for service registration.
Domain Organization	API takes the specific request, Verification of authorization.	If organization authorized, API takes request and returns results as expected

6.2.2 Test Schedule

Here is our scheduling for sign language recognition & text to sign puppet model testing including the web & mobile application –

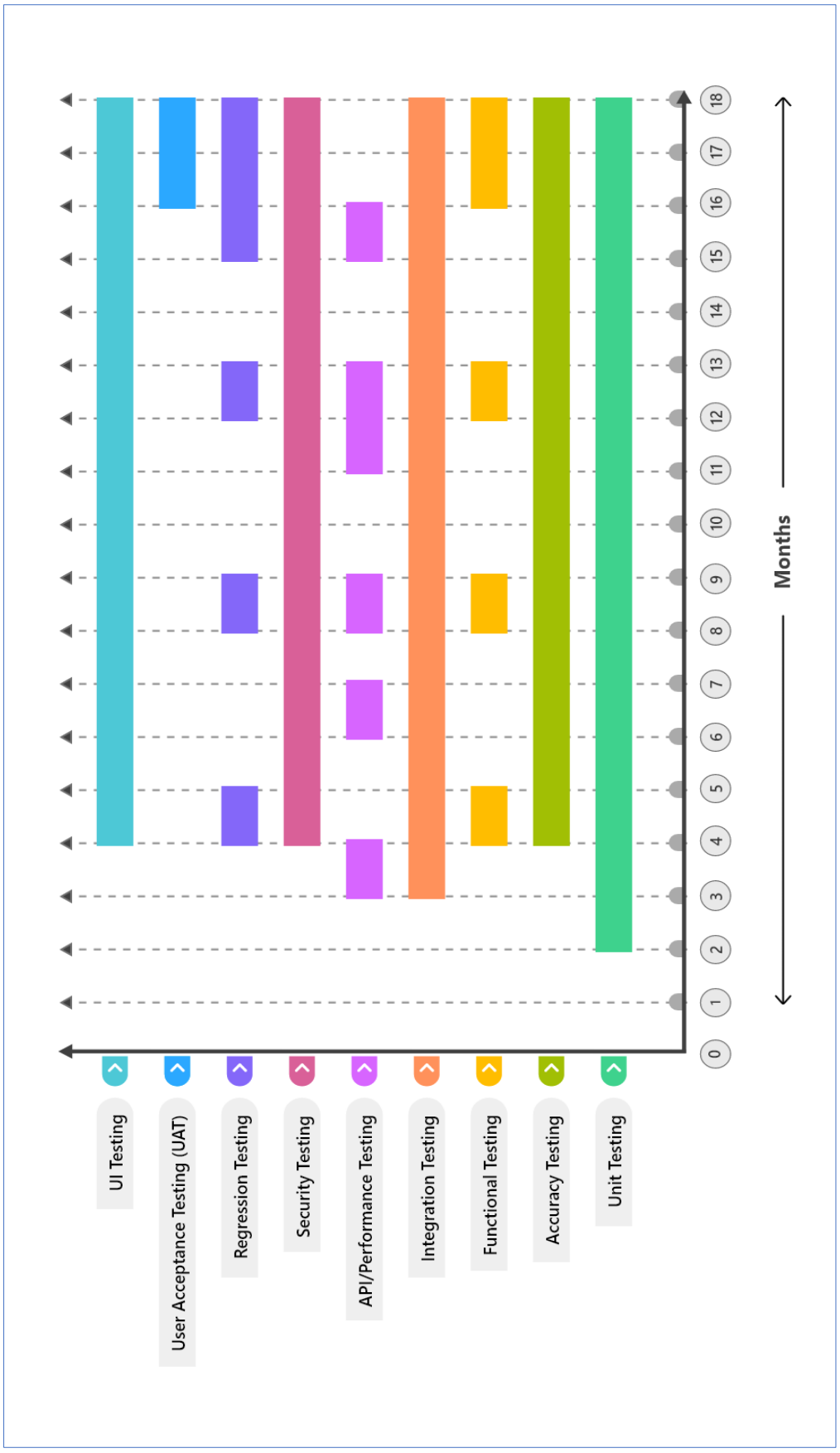

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Months
UI Testing
User Acceptance Testing (UAT)
Regression Testing
Security Testing
API/Performance Testing
Integration Testing
Functional Testing
Accuracy Testing
Unit Testing

Figure 99. Test Schedule for SLR & T2SP

Below timeline is our scheduling for Bangla to Braille converter with Editor & its Office Add-ons testing –

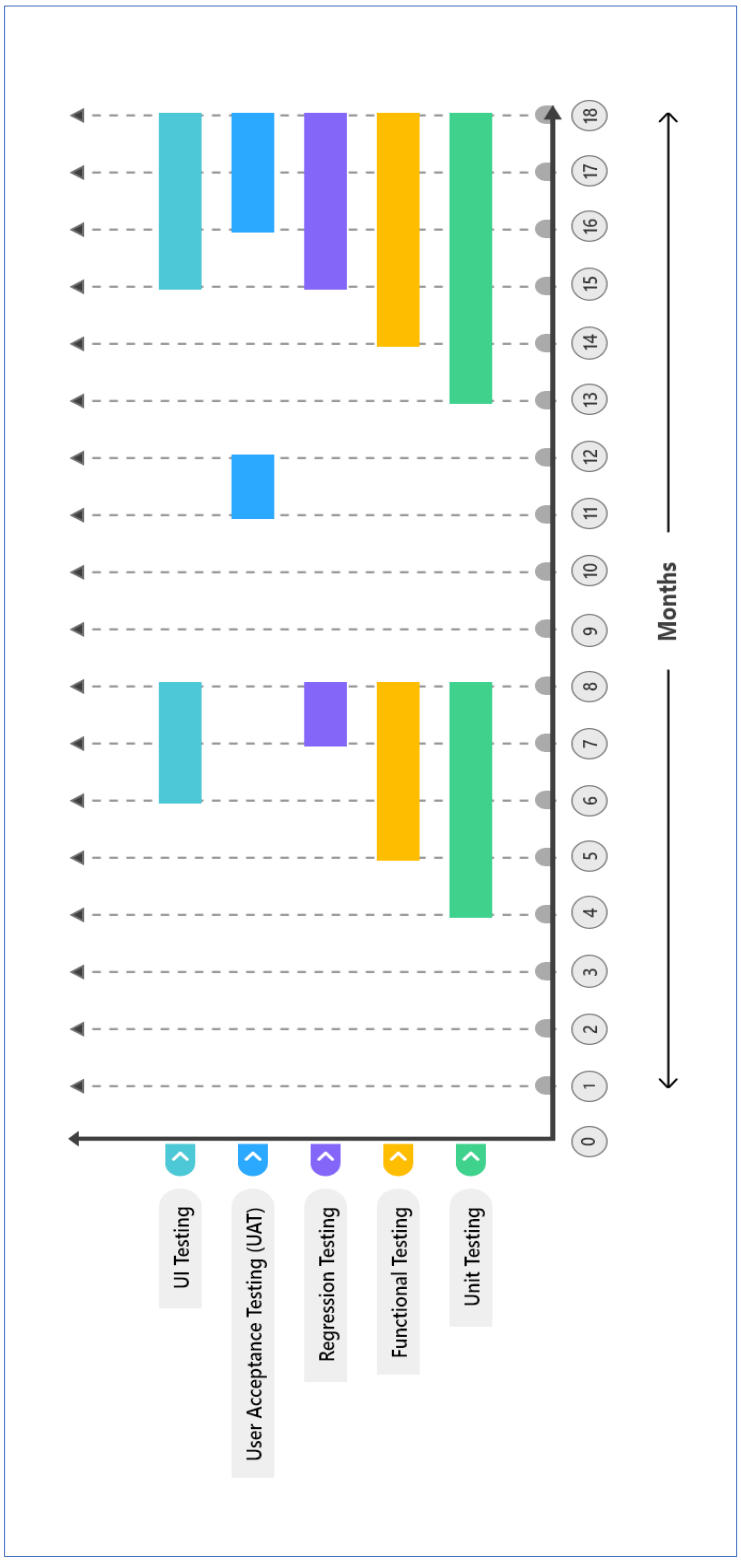


Figure 100. Test Schedule for B2BC Converter & Office add-ons

6.3 Risk Management

The development of an application for automatically and robustly translating between Bangla Sign Language and Bangla Unicode Text and Text to signing avatar in real time on computer/devices holds the promise of enabling natural and seamless communication between deaf signers and normal people anywhere and anytime. One key component of such a system is the automatic recognition of Bangla signs, which is an active area of research and investigation of this project.

A good number of major risks factors and challenges are there to achieve the optimal solutions:

6.3.1 Risk in the ground:

Sign Standards: Very limited standardized gloss signs are there in the deaf community, from there very few people follow that standard. Preparing the video dataset maintaining the standard won't be a problem but getting the expected accuracy from rest of the glosses which hasn't been standardized would be an issue as dialect will come into the picture. Some will consider it right and others will consider it wrong in the deaf community creating a huge risk.

Institutionalization: Society of the Deaf and Sign Language Users (SDSL) is the lone organization attempting to teach sign language using an organized framework in In Dhaka, Bangladesh. BdSL is not yet institutionalized in the broad spectrum and hence teaching and learning capacity is still narrow and negligible. Even though the country has more than 30 lakh hearing impaired people, the opportunities for learning sign language is inadequate and disappointing.

Growing Experts: Sign language has been using in news translation in (BTV), in international conference and seminars – but very few programs are considered to broadcast for the sake of deaf community. As there is no designated institutions and no department is yet introduced in national university level - there is a dearth of teachers and trainers who could multiply the number of accurate signers by following a minimum standard.

Limited Access: Bangla Sign Language is recognized as a language of Bangladesh, it still does not have any legal protection and no steps have been taken towards institutionalization of the language. As a result sign language users can not enjoy full access to vital information and services including education, health services and employment opportunities.

Research and Development: Very few and limited researches have been conducted which is justified not to enrich the Sign Language community, though the rapid development has been noticed in the technological area – no such application is seen for deaf community that helps them to contribute to the national development. Conducting extensive research and study would be possible only if we have a language institute for Bangla sign language under the guidance of the Ministry of Education and governments considerations.

6.3.2 Technological Risk for BdSLR:

Apart from risks in the ground, for this particular application we have technological challenges as well. First of all let's set an examples, for the following applications for American Sign Language, they had lot of researches, corpus data and test data set for training machine:

- SignAll (http://www.signall.us/)
  First they developed a prototype and then they came for the commercial market, though it is not yet available for purchase, and few technical details are known beyond the fact that it uses three stationary standard video cameras, a depth camera, and a personal computer.

- KinTrans (https://www.kintrans.com/)
  This is a Kinect-based system being developed for two-way communication between English speaking and American Sign Language signing individuals.

- Microsoft (https://www.microsoft.com/en-us/research/blog/kinect-sign-language-translator-part-1/)
  This giant was actively involved with developing a Kinect-based ASL translator (Chai et al 2013, Chen et al 2013) with its Chinese academic partners in 2013. There is no evidence that Microsoft has actively pursued this area since then, though their Chinese academic partners from this earlier work have continued to publish in this technical area.

- MotionSavvy (http://www.motionsavvy.com) [the link is old one]
  It's a tablet based app that understands sign language - is ostensibly moving toward a commercial release of its tablet-based system that leverages a Leap Motion detector (two visible and three infrared sensors). The Leap Motion-based system does support a level of mobility that is not available with the other discussed systems.

All of the companies/organizations listed above spent several years to achieve the sign language translation with the help of Artificial Intelligence. Still they are not up to the mark. But in our case, it's rarely seen a successful research has been done, any organization has come in this field to invest so risk is high for this initiative – but genweb2 has taken challenges to reach to close to fare solution.

We see the following challenges may arise in the course the development of the applications:

Risk in hypothesis based operation: As we are still based on hypothesis meaning that we have to find a way where we can get the maximum accuracy for the targeted problem. So we will have variations on what model we want to use or what pipeline we want to build that will not meet the way of thinking the data collection or our findings will not meet with the real world end users need until we get a standard acceptable predictions – we are doing some RnD so that we can come to close and accurate translation.

Data alignment: The most significant factors are the size of input data; the presence of noise, outliers and occlusions; the quality of the extracted features; real-time requirements; and the type of transformation, especially those defined by multiple parameters. After collecting video data, extracting frames from video and having augmentation and annotation is huge challenge for massive data - we are planning them to mitigate with minimal approaches.

Wrong annotation or Poor annotation: Any machine learning model can only be as good as the training data that was used. If we don't annotate all occurrences of a field, the model will be confused when it needs to extract this field (or not) and give low confidence scores. For this particular problem, a huge bunch of video data is there and we need to extract frames from videos and annotate them appropriately. A close review and observation is needed to make sure every field has been annotated accurately and properly, otherwise it will be a source of error to make the model bias and erroneous.

Inaccurate Augmentation Process: Data augmentation techniques have become standard practice in deep learning, as it has been shown to greatly improve the generalization abilities of models. These techniques rely on different ideas such as invariance-preserving transformations (e.g. expert-defined augmentation), statistical heuristics (e.g. Mix-up), and learning the data distribution (e.g., GANs). However, in the adversarial settings it remains unclear under what conditions such data augmentation methods reduce or even worsen the misclassification risk. To consider accurate augmentation technique - we can consider already proven techniques like Image Shifting, Image Flipping, Image Noising, and Image Blurring and so on.

Inappropriate Model Biasness: Over or under training results biasness in the model, machine gets confused what to do then. If the training data is incomplete or does not provide a proper balance in the range of data supplied, then the neural net will be trained in a similarly skewed manner. Model performance mismatch leads to incorrect decisions with a high risk of bias. Obviously, we need a way to detect bias, and upon detection, remove or negate it. While this isn't easy, you can take a few actions to lessen the impact of bias. Prevention better than cure - we can follow the below techniques to avoid biasness:

- Choose the correct learning model
- Use the right training dataset
- Perform data processing mindfully
- Monitor real-world performance across the ML lifecycle
- Make sure that there are no infrastructural issues

6.3.3 Risk Prioritization

Risk prioritization is the ordering of identified risks from the highest to the lowest level of risk exposure to the project. In BdSLR project risks will is sorted by category and assessed as being either negotiable or uncontrollable. They are then ordered from highest to lowest risk, based on the risk exposure and we need to mitigate those by monitoring & control mechanism.

6.3.4 Monitoring & Control Mechanism

The key to risk monitoring & control mechanism in risk management is understanding all the potential risks to the project and ensuring that these potential risks and risk mitigation strategies are communicated to key project stakeholders on an ongoing basis. Some of our monitoring and mitigation action plan for risk management for this project is included below.

SL	Risks	Impact	Mitigation Strategy
1	We need to train our models using machine learning algorithms which requires high performing hardware resources.	Extended time needed and the performance of our model may degrade.	Inform stakeholder to increase the hardware as soon as possible
2	The methodologies which we intend to follow as they work well for other languages may not work as assumed for Bangla.	We may need more time than we expected.	We will keep track of progress and inform the stakeholder informing we need researches do get the right one.
3	Proper data augmentation process was not considered initially and also latest augmentation process came in.	The application may not reflect proper accuracy having no much data in the defined model.	A quick back-end model need to keep read so that experiment can be applied on that model.
4	We need to integrate APIs from corpus development tool but API is not available at that end.	If the API's not available, a delay is there to integrate API.	Will be handled upon mutual agreement.
5	Crowdsourcing sign is not annotated properly.	Real time sing will not give close accurate conversion to text.	Inform stakeholders to develop a process to annotated crowd sourced sign.

7. Technology Stack

7.1 Web Based SLR

Following are the technologies we will use for the development of SLR

- **SLR Technologies**
    - **Language:** Python 3 or above
    - **Distribution:** Docker, NVIDIA-Docker
    - **Engine:** TensorFlow, PyTorch, CUDA, NLTK, GStreamer, imagemq, TensorBoard, Transformer, YOLOv5, Detectron2, Apache-Kafka, BERT, DensePose, FaceMesh
    - **Database:** Elasticsearch, MongoDB, Neo4j, PostgreSQL, Redis

- **Web SLR Technologies**
    - **Front-End:** HTML, CSS, JavaScript
    - **Front-End JS Framework:** Angular, TypeScript
    - **Front-End CSS Library:** Bootstrap
    - **Back-End:** Java
    - **Database:** MySQL

7.2 Web Based T2SP

Following are the technologies we will use for the development of T2SP

- **T2SP Technologies**
  - **Language:** Python 3 or above
  - **Distribution:** Docker, NVIDIA-Docker
  - **Engine:** TensorFlow, PyTorch, CUDA, NLTK, GStreamer, spaCy, DCGAN, GAN, flairNLP, imagemq, TensorBoard, Transformer, , Apache-Kafka, BERT
  - **Rendering technology:** X3D, WebGL
  - **Database:** Elasticsearch, MongoDB, Neo4j, PostgreSQL, Redis

- **Web T2SP Technologies**
  - **Front-End:** HTML, CSS, JavaScript
  - **Front-End JS Framework**: Angular, TypeScript
  - **Front-End CSS Library**: Bootstrap
  - **Back-End:** Java
  - **Database**: MySQL

7.3 Technology stack for Desktop Applications

- Electron - platform independent IDE, computable for windows, Unix/Linux and mac
- Node Package Manager (NPM)
- NodeJS
- WebHook
- WebRTCj

7.4 BdSL Enabled Web Based Communicator

- **Front-end:** HTML, CSS, JS, Angular, Typescript
- **Back-end:** Java/ Erlang/ Golang
- **Message-Broker:** Apache Kafka/ RabbitMQ
- **Database:** Mysql / MongoDB
- **DevOps:** Docker, Kubernetes

7.5 SLR Mobile Application

- **Android Platform Technologies**
  - **Language:** Java 8 or above
  - **Android Application Type:** Native Android App using Google's Android SDK

- **UI Language/Tool:** XML, Java
- **Local Database:** Realm, Sqlite, built-in storage (Shared Preference)
- **Network Connectivity tool:** Retrofit 2

- **iOS Platform Technologies**
  - **Language:** Swift
  - **iOS Application Type**: Native IOS App using Apple IOS SDK
  - **UI Language/Tool:** Interface Builder (Auto Layout)
  - **Local Database:** Realm,CoreData
  - **Network Connectivity tool:** AFNetworking

7.6 SLR Bangla to Braille Converter

- **Development Language:** JAVA
- **Web App Framework:** Spring (Web MVC)
- **Persistence Framework:** MyBatis
- **Presentation Framework:** Angular, jQuery, HTML5, Bootstrap CSS
- **Web Application Server:** Apache Tomcat
- **Database Server:** MySQL

7.7 Hardware

Configuration

When trying to gain business value through machine learning, access to best hardware that supports all the complex functions is of utmost importance. With a variety of CPUs, GPUs, TPUs, and ASICs, choosing the right hardware may get a little confusing. So, for building an infrastructure for machine learning projects need extra care best on the processing of everything we want to deal. Machine learning is basically a mathematical and probabilistic model which requires tons of computations. It is very trivial for humans to do those tasks, but computational machines can perform similar tasks very easily. Consumer hardware may not be able to do extensive computations very quickly as a model may require calculating and update millions of parameters in run-time for a single iterative model like deep neural networks.

There are four steps for preparing a machine learning model:

- Preprocessing input data
- Training the deep learning model
- Storing the trained deep learning model
- Deployment of the model

Among all these, training the machine learning model is the most computationally intensive task.
All in all, while it is technically possible to do Deep Learning with a CPU, for any real results you should be using a GPU. So, GPU consideration we need think twice, for optimal performance a good spec of GPU needs to be selected.

What to look for in a GPU?

There are main characteristics of a GPU related to Deep Learning are:

Memory bandwidth: The ability of the GPU to handle large amount of data. The most important performance metric.

Processing Power: Indicates how fast your GPU can crunch data. We will compute this as the number of CUDA cores multiplied by the clock speed of each core.

Video RAM Size: The amount of data you can have on the video card at once.

CPU Vs. GPU

- First, we need to learn that CPUs are latency optimized while GPUs are bandwidth optimized.
- The CPU is good at fetching small amounts of memory quickly (5 * 3 * 7) while the GPU is good at fetching large amounts of memory (Matrix multiplication: (A*B) *C).
- The best CPUs have about 50GB/s while the best GPUs have 750GB/s memory bandwidth. So, the more memory your computational operations require, the more significant the advantage of GPUs over CPUs.
- GPUs offer high bandwidth while hiding their latency under thread parallelism, so for large chunks of memory GPUs provide the best memory bandwidth while having almost no drawback due to latency via thread parallelism. This is the second reason why GPUs are faster than CPUs for deep learning.
- The advantage of the GPU is here that it can have a small pack of registers for every processing unit (stream processor, or SM), of which it has many. Thus, we can have in total a lot of register memory, which is very small and thus very fast. This leads to the aggregate GPU registers size being more than 30 times larger compared to CPUs and still twice as fast which translates to up to 14MB register memory that operates at a whopping 80TB/s.

If the task is of a larger scale than usual, GPU cluster can be considered along with multi-GPU computing. Also, there are more powerful options available – TPUs and faster FPGAs which are designed specifically for these purposes. Considering the performance and accuracy of deep learning application like SLR, we may need to tune the hardware on demand basis, if training dataset becomes a factor of subsequent modification of the application.

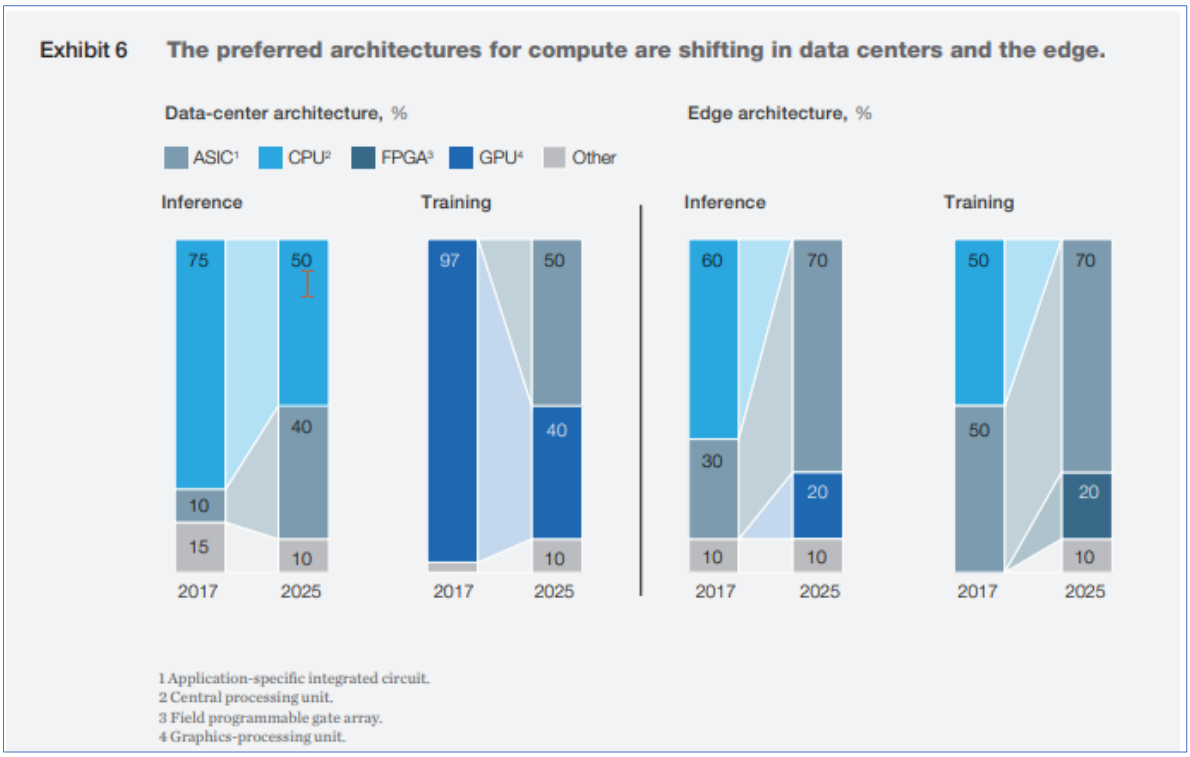


Figure 101. AI Hardware

For Server

Hardware	Description	Details
GPU	GPU can perform convolutional/CNN or recurrent neural networks/RNN based operations. It can also perform operations on a batch of images of 128 or 256 images at once in just a few milliseconds. If one GPU is not enough, the concept of multi-GPU computing may need to be introduced.	PCI Express RTX 3080 Ti • 12GB DDR6 • 320 Tensor Core • 10,240 GPU Core • 80 RT Cores • 912 GB/s Memory Bandwidth
CPU	The CPU does little computation when we run our deep nets on a GPU. Mostly it (1) initiates GPU function calls, (2) executes CPU functions.	Intel® Xeon® Silver 4214R Processor, 16.5 MB Cache, 12 Cores, 24 Threads 3.50 GHz Max Turbo Frequency
SSD	An SSD for productivity: Programs start and respond more quickly and pre-processing with large files is quite a bit faster. For smooth operation choice can be on NVMe SSD.	SSD M.2 NVMe 1 TB Capacity
HDD	An HDD for storage: For different file storage a HDD can be used.	HDD SATA on RAID 2 TB

Motherboard	Motherboard is the main printed circuit board (PCB) in general-purpose computers and other expandable systems	Intel Turbo Boost Technology, LGA-3647 Socket, 14nm, 85w TDP] EP2C621D12 WS Workstation Board Dual Socket
RAM	Random-access memory (RAM) is a form of computer memory that can be read and changed in any order, typically used to store working data and machine code.	DDR4 ECC, 64GB, 2933/2666MHz Server Memory
Power Supply	Power supply amount need to be calculated with the required watts by adding up the watt of the system CPU and GPUs with an additional 10% of watts for other components and as a buffer for power spikes.	1250watt Goldreal rated Server Grade PSU
Monitor	To monitor hardware resource consumption along with power, a general-purpose monitor will suffice.	Dell E2417H 24 Inch LED Monitor

For Workstation

Hardware	Description	Details
GPU	GPU can perform convolutional/CNN or recurrent neural networks/RNN based operations. It can also perform operations on a batch of images of 128 or 256 images at once in just a few milliseconds. If one GPU is not enough, the concept of multi-GPU computing may need to be introduced.	GeForce GTX 1050 Ti • 4GB GDDR5 • 768 GPU Core • 1,392 MHz Boost Clock
CPU	The CPU does little computation when we run our deep nets on a GPU. Mostly it (1) initiates GPU function calls, (2) executes CPU functions.	Intel core i7, 11th Gen, 11700
SSD	An SSD for productivity: Programs start and respond more quickly and pre-processing with large files is quite a bit faster. For smooth operation choice can be on NVMe SSD.	SSD M.2 NVMe 1 TB Capacity, 220 S
HDD	An HDD for storage: For different file storage a HDD can be used.	N/A

Motherboard	Motherboard is the main printed circuit board (PCB) in general-purpose computers and other expandable systems	MPG Z590 Gaming Edge WiFi
RAM	Random-access memory (RAM) is a form of computer memory that can be read and changed in any order, typically used to store working data and machine code.	G Skill Trident 32GB, DDR-4, 3200 Bus
Power Supply	Power supply amount need to be calculated with the required watts by adding up the watt of the system CPU and GPUs with an additional 10% of watts for other components and as a buffer for power spikes.	Thermaltake Toughpower GF1 750W 80 Plus Gold Fully
Monitor	To monitor hardware resource consumption along with power, a general-purpose monitor will suffice.	HP M22F 22" FHD

Note: This recommendation goes for running an application that is targeted on ML/DL based for smoother performance. For Private-Cloud data center, specifications need to be reviewed and adjusted aligning hardware in the datacenter.

8. Management, Organization & Staffing

8.1 Team-Composition-for-SLR

8.1.1 Team-Composition-for-SLR

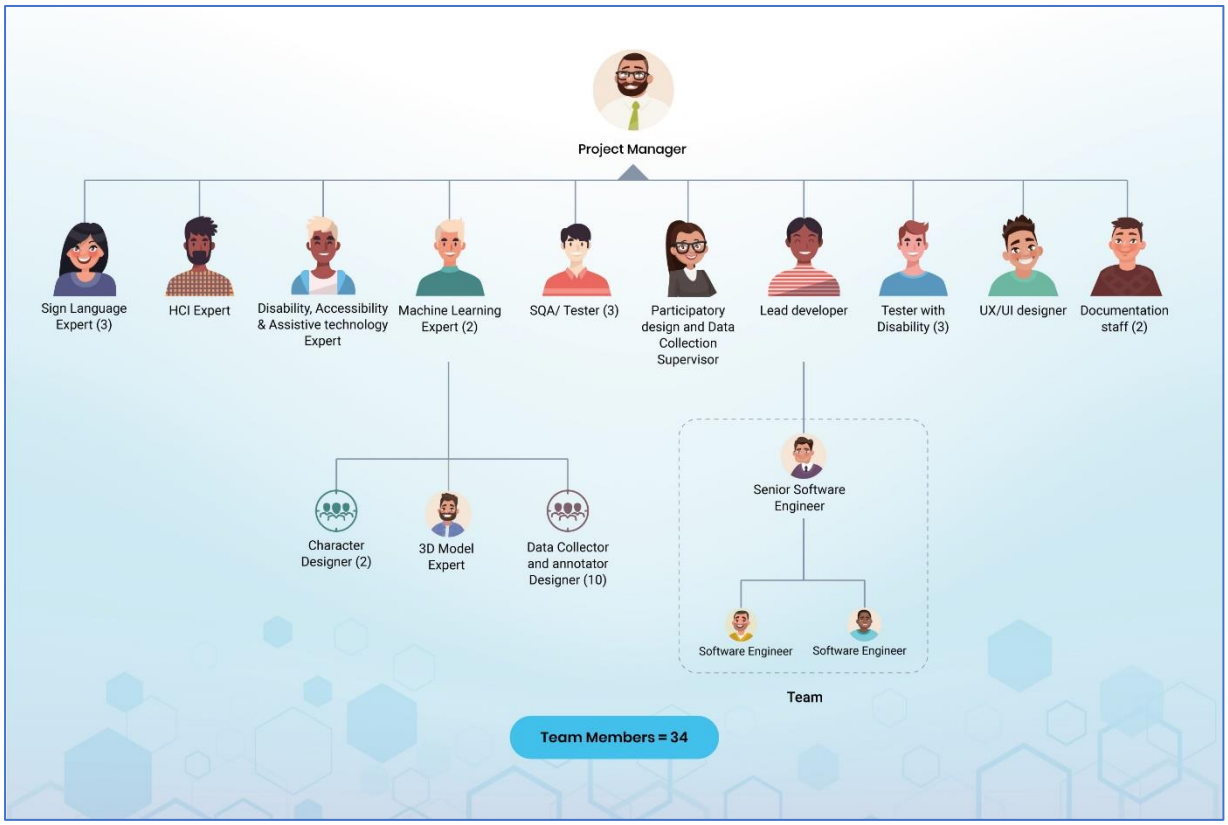


Figure 102. Team-Composition-for-SLR

8.1.2 Team-Composition-for-T2SP

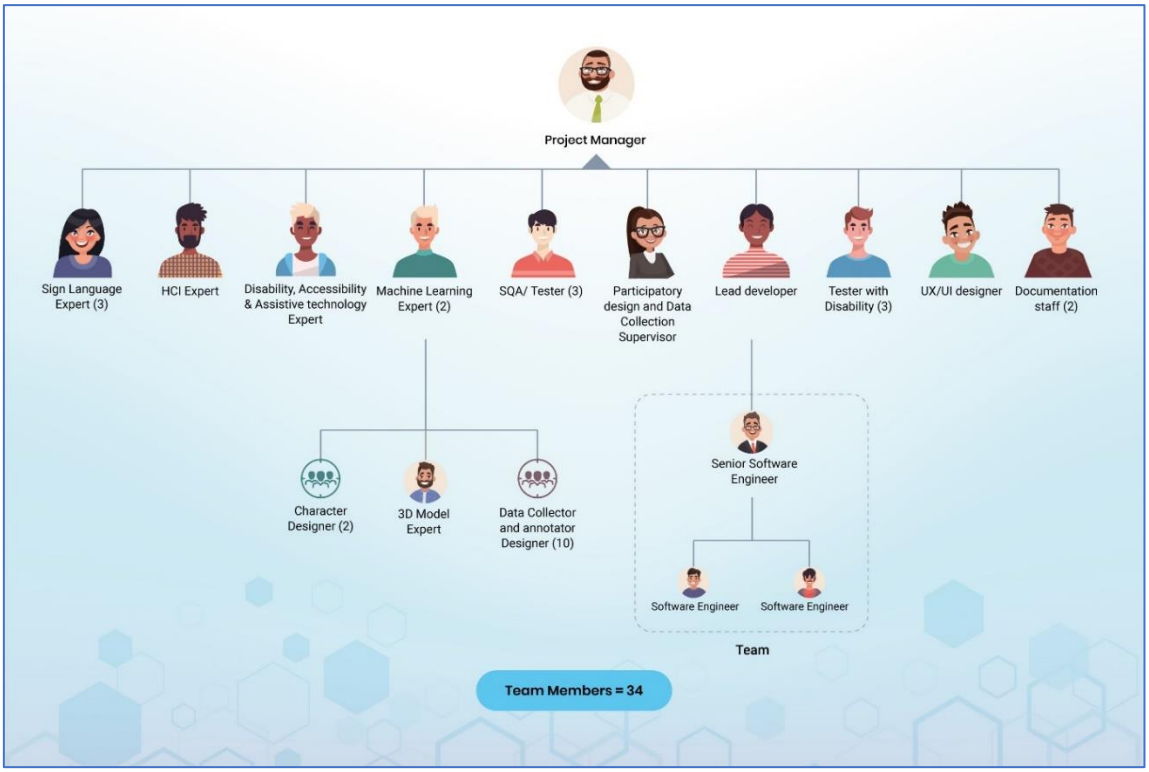


Figure 103. Team-Composition-for-T2SP

8.1.3 Team-Composition-for-Braille

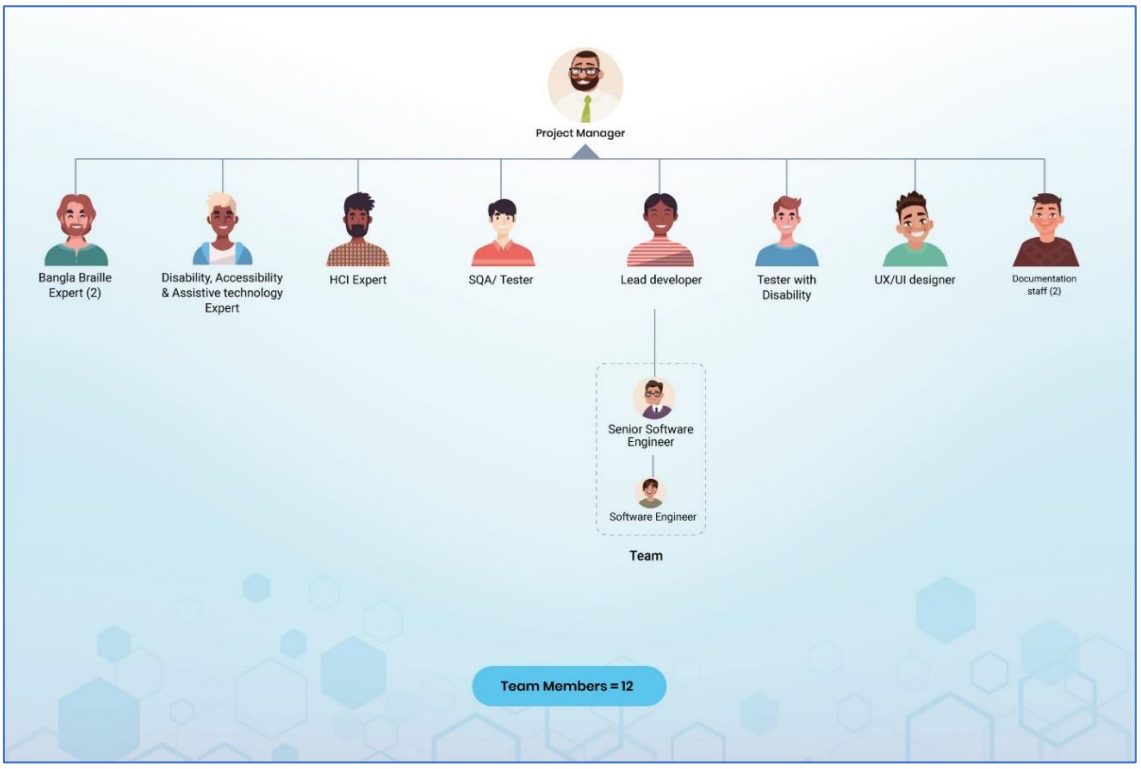


Figure 104. Team-Composition-for-Braille

8.2 Manpower Requirement for SD-17 Project

According to team distribution chart, we would be requiring following manpower to complete the development phase.

Designation	Man Count
Team leader/manager	1
Sign language Expert	3
Bangla Braille Expert	2
Disability, Accessibility and Assistive technology Expert	1
Participatory design and Data Collection Supervisor	2
Machine Learning Expert	4
HCI Expert	1
Character Designer	2
3d Model Expert	1
Lead developer/Technical lead/Software architect	1
Senior software engineer	2
Software engineer	4
SQA/ Tester	4
Tester with Disability	4
Data Collector and annotator Designer	10
UX/UI designer	1
Documentation staff	1
Total	**44**

Team Composition for SLR

Name of Staff	Position Assigned
Shahidul Haque	Team leader/Project Manager
Arafat Sultana	Sign Language Expert
Bodrunnaher Toma	Sign Language Expert
Ariful Islam	Sign Language Expert
Rafiul Alam	Disability, Accessibility and Assistive technology Expert
Sadia Afrose	Participatory design and Data Collection Supervisor
Shahidul Islam	Machine Learning Expert
Ashraful Alam	Machine Learning Expert
Md Iqbal Mahmud	HCI Expert
Shahin Mahmud	Character Designer
Samia Akter	Character Designer
Majedur Rahman	3D Model Expert
Jobaidul Alam Jamali	Lead developer/Technical lead/Software architect
Pranab Sarker	Senior Software Engineer
Abu Rayhan	Software Engineer
Tahmidul Abedin	Software Engineer
Faisal Mahtab Khan	SQA/ Tester
Oveget Das	SQA/ Tester
Rashedul Alam	SQA/ Tester
Md. Jahangir Alam	Tester with Disability
Summaiya Ferdousi Munny	Tester with Disability
Md. Abdullah	Tester with Disability
Dola Chakroborty	Data Collector and annotator Designer
Farzana Sultana	Data Collector and annotator Designer
Golam Rabbi	Data Collector and annotator Designer
Ahad Ahmed	Data Collector and annotator Designer
Warif Akand Rishi	Data Collector and annotator Designer
Zubair Raihan	Data Collector and annotator Designer
Shoaib Ahmed	Data Collector and annotator Designer
Aynul Islam	Data Collector and annotator Designer
Sumaiya Nasrin	Data Collector and annotator Designer
Farhanul Islam	Data Collector and annotator Designer
Moinuddin	UX/UI designer
Hossain Al Nahhyan	Documentation staff
Umma Kulsum Rita	Documentation staff

Team Composition for T2SP

Name of Staff	Position Assigned
Shahidul Haque	Team leader/Project Manager
Arafat Sultana	Sign Language Expert
Bodrunnaher Toma	Sign Language Expert
Ariful Islam	Sign Language Expert
Rafiul Alam	Disability, Accessibility and Assistive technology Expert
Sadia Afrose	Participatory design and Data Collection Supervisor
Mahmudul Hassan Navid	Machine Learning Expert
Mehedi Hasan Tushar	Machine Learning Expert
Md Iqbal Mahmud	HCI Expert
Shahin Mahmud	Character Designer
Samia Akter	Character Designer
Majedur Rahman	3D Model Expert
Jobaidul Alam Jamali	Lead developer/Technical lead/Software architect
Md. Mahfuzur Rahman	Senior Software Engineer
Ratan Dhar	Software Engineer
Mehedi Hasan	Software Engineer
Faisal Mahtab Khan	SQA/ Tester
Oveget Das	SQA/ Tester
Rashedul Alam	SQA/ Tester
Md. Arif Hosen	Tester with Disability
Summaiya Ferdousi Munny	Tester with Disability
Md. Abdullah	Tester with Disability
Dola Chakroborty	Data Collector and annotator Designer
Farzana Sultana	Data Collector and annotator Designer
Golam Rabbi	Data Collector and annotator Designer

Ahad Ahmed	Data Collector and annotator Designer
Warif Akand Rishi	Data Collector and annotator Designer
Zubair Raihan	Data Collector and annotator Designer
Shoaib Ahmed	Data Collector and annotator Designer
Aynul Islam	Data Collector and annotator Designer
Sumaiya Nasrin	Data Collector and annotator Designer
Farhanul Islam	Data Collector and annotator Designer
Moinuddin	UX/UI designer
Hossain Al Nahhyan	Documentation staff
Umma Kulsum Rita	Documentation staff

Team Composition for Braille

Name of Staff	Position Assigned
Shahidul Haque	Team leader/Project Manager
Md. Abdus Samad	Bangla Braille Expert
Md. Rashedujjaman Chowdhury	Bangla Braille Expert
Rafiul Alam	Disability, Accessibility and Assistive technology Expert
Jobaidul Alam Jamali	Lead developer/Technical lead/Software architect
Md Iqbal Mahmud	HCI Expert
Md. Mahfuzur Rahman	Senior Software Engineer
Abu Rayhan	Software Engineer
Raj Narayan Saha	SQA/ Tester
Md. Arif Hosen	Tester with Disability
Moinuddin	UX/UI designer
Hossain Al Nahhyan	Documentation staff
Umma Kulsum Rita	Documentation staff

9. Change Management & Version Control

91 Change Control

Change management is the whole process of change request & control and this needs to maintain whenever a new or different changes are requested for this system. It must be implemented without affecting other components of this system. It will help us, project teams to modify the scope of the project using specified controls and policies.

We will follow mandatorily different formal document for change request completion and revision in order to keep control of change requests.

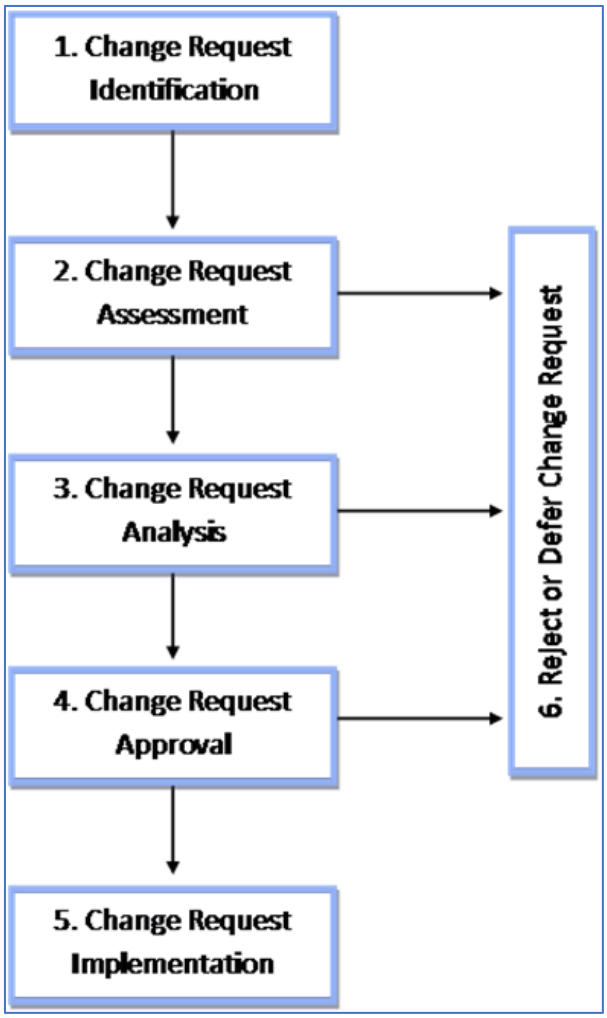


Figure 105. Change Management

Steps for Change Control	Action
Change request identification	• Identify the need for a change and describe it on the project change request form.
Change request assessment	• If the change is not valid, it must be deferred or rejected. Determine appropriate resources required to analyze the change request. • Perform quick assessment of the potential impact and update the change request form. • At this stage, rejected change request should stopped
Change request analysis	• For analysis assign the change request to an authorized member. • Deferred change re-enter this analysis step. • At this stage, rejected change request should stopped.
Change request approval	• Identify change risk and complexity level before approval. • Identify the impact level of the change before approval. • Review impact of Change Request to authorized person for approval. • At this stage, rejected change request should stopped.
Change request implementation	• Update project procedure and management plans. • Inform about the changes to the team. • Monitor progress of change request. • Record the completion of change request. • Close change request.

9.2 Version Control

It's also known as revision control which is a version management strategy to track and store changes to a software development document or set of files that follow the development project from beginning to end-of-life. Following the logical and recorded development trail in our SD-17 project, this process can aid in finding problem origins, bugs, or even better versions of the build. Version Control is a data repository (or repositories) that will track, every change made to a file, who made the change, and when the change occurred. This information will track all input to create a reliable historical record of this entire project. There is a different category of Version control systems and in between them we will move with the **Distributed Version Control System.**

Version controlling will give us some key benefit with the SD-17 project –

- Organized, coordinated management of changes to these applications assets for emergency hot-fixes, routine maintenance, upgrades, and new features with potentially overlapping development timeframes (e.g., work on new features occurs simultaneously with work on routine maintenance and/or hot-fixes)
- An auditable change history (e.g., what changed, when, and by whom)
- A reliable master copy of what assets are currently in production
- A reliable master copy of assets from which to build and/or configure the production environment

- Reliable copies of previous production versions of assets
- Ability to see the specific differences between distinct versions of a given asset

Why Distributed Version Control System?

Distributed open and closed sourced version control systems have different functional elements for collaborative editing and sharing. [5] A key element is that there is not one centralized repository for change information. Instead, each developer has their own historical record and their own repository. This approach is often credited with moving a project along faster than its centralized counterparts. Distributed version control systems (DVCS) allow each developer to work out challenges before adding to the master document in the main repository; these multiple "working copies" are in process for eventual merge or rejection. This style is often cited as a good system for large-scale projects like EBLICT project. Like centralized system, DVCS works to avoid the problems of stepping on or overwriting others' work and provides document history preservation and reversion capabilities. [5]

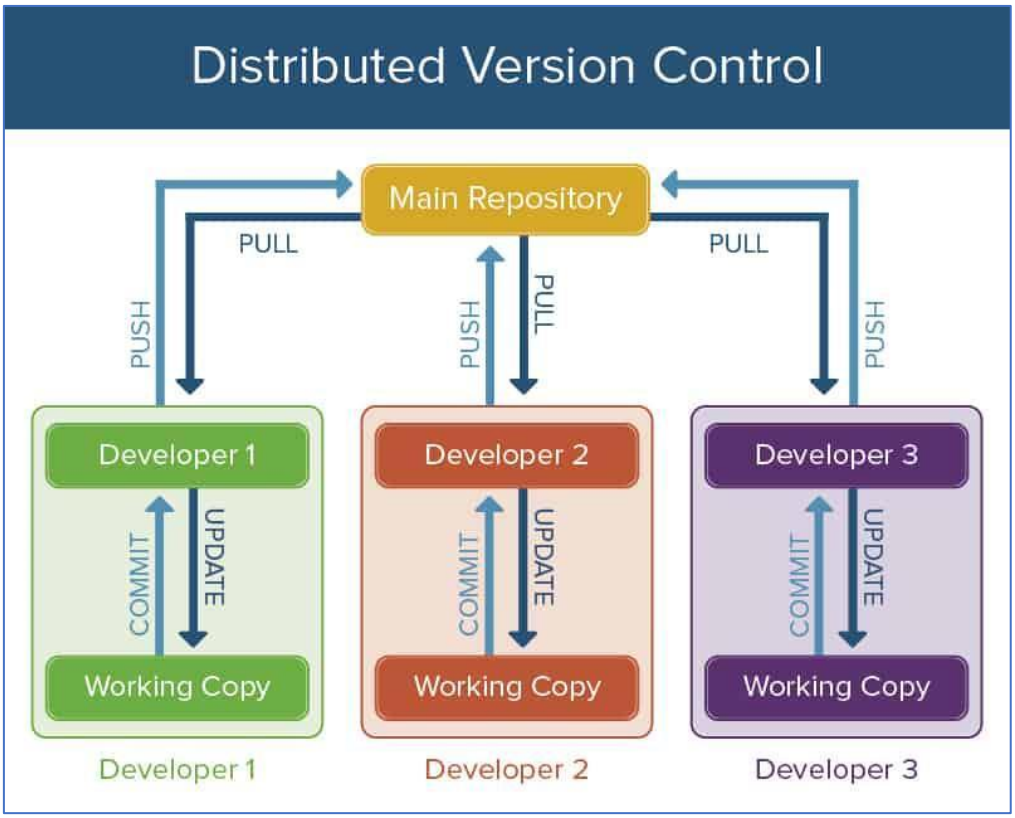


Figure 106. Distributed Version Control Process

10. Maintenance

10.1 Application & apps Maintenance

The life of a software/application does not begin when coding starts and end with the launch. Instead, it has an ongoing lifecycle that stops and starts whenever necessary. The start of its lifecycle and a good portion of the work begins at launch. Software/application is always changing and as long as it is being used, it has to be monitored and maintained properly. This is partly to adjust for the changes within an organization but is even more important because technology keeps changing.

All the "SD 17" components like SLR, T2SP & B2BC applications may need maintenance for any number of reasons – to keep it up and running, to enhance features, to rework the system for changes into the future, to move to the Cloud, or any other changes.

There are four types of software/application maintenance-

- Corrective Software Maintenance
- Adaptive Software Maintenance
- Perfective Software Maintenance
- Preventive Software Maintenance

Rather than all these 4 types of maintenance, we need to workover on some more areas of maintenance for "SD 17" components.

Documentation: There will be a plan as documented for overall operational by sub component wise. Since these applications will be used mostly by the mass people, we will provide very well designed user manual for installation and proper use of those applications. Those documents will help other secondary instructor also to maintain the knowledge flow between the users and different organizations those who are connected with "SD 17" projects. It will be elaborately written in the document that how the desktop SLR application needs to install in different OS like Windows, UNIX and Mac. There will be written clearly about the download and installation process with the link of Android play store and iOS apps store for SLR & Braille application. None the less the manual will help the user to understand that how to use those application seamlessly including the video tutorial if there is any complex operations. Rather than manuals, we will provide API documentation and technical documentation of SLR, T2SP & B2BC.

There will also be separate manuals for the administrative users of the project maintenance team who will augment corpus and make enhancement further.

Scalability Plan: Any application can be considered as scalable if there is no need to re-design it to maintain effective performance during or after a steep increase in workload. All of the users those who have hearing impairment disability can communicate seamlessly through SLR system with other normal users. In this SLR system, a DL Model is active and functional behind the application which is helping to identify and recognize Bangla sign language and convert

into Bangla text and by T2SP system, it can convert the text to Bangla sign language with a visual puppet activity. In this case, we can say that, SLR system is scalable if we don't need to re-design the system or don't need to maintain temporarily downtime to update or upgrade the DL Model.

By the flow of time, it may need to improve the DL model by providing it new learning materials and it will fall under maintenance or further development, once the SLR system have been finally implemented and already using by end users. There is 2 ways that we need to move with. If introducing a new feature with the capabilities of existing DL Model, we can simply develop the backend in dev mode and after the feature tested, we can just add the feature with existing application. But if we need a new feature which needs the DL Model to train further then we will use **Docker** Platform. 1st let's get introduce with Docker a little –

"Docker is an open platform for developing, shipping, and running applications. Docker enables us to separate our applications from our infrastructure so we can deliver software quickly without any downtime. Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so we do not need to rely on what is currently installed on the host.
Docker provides tooling and a platform to manage the lifecycle of containers:

- Develop/Update our application and its supporting components using containers.
- The container becomes the unit for distributing and testing our application.
- When we're ready, can deploy our application into our production environment, as a container."

Now about using, connecting and updating the DL Model of SLR through Docker platform is very clear conception. In case of any new feature addition with additional training to existing DL Model is not possible. In this case we need to train the existing model with additional training and after successful testing, we can set the new Model into a container B of Docker where existing Model is in another container A. After that we will just introduce the new container B to backend to be functional. These whole process doesn't need any downtime at all.

Escalation Matrix & Issue Categorization: An escalation matrix is a system that defines when escalation should happen and who should handle incidents at each escalation level.

The below table will make us clear that after a successful implementation of The SLR with T2SP and B2BC application, who is going to take action for which of the following category of issues. It also mention that which category issue will be on what exact level.

Level	Person responsible	Role	Triggers when	Issue Category
Level 6	Md. Obydul Islam	GM	• Overall product issue.	Product (Moderate, Critical)
Level 5	M. Shahidul Haque	Project Manager	• SLR, T2SP & B2BC operational issues.	Functional, Performance (Critical)
Level 4	Md. Iqbal Mahmud	System Engineer	• Network security issue. • SLR System Unavailability. • Server data backup/restore issue.	System, Security (General, Critical)
Level 3	Jobaidul Alam Jamali	Technical Lead	• Overall functional issue • Feature modification/alteration. • Application security issue.	Functional, Performance, Security (Moderate)
Level 2	Shahidul Islam	Engineer	• SLR web, desktop application load issue. • SLR, Braille mobile apps buffering issue. • Data unavailability issue. • Error messages handling. • Workflow problem.	Functional, Performance (Minor)
Level 1		Support	• General user help/query. • Device & apps connection issue.	General

Table 15. Escalation Matrix & Issue Categorization

There will be a plan as documented for overall operational by sub component wise.

Root Cause Analysis: Root Cause Analysis (RCA) is a maintenance troubleshooting method that will helps us to identify and control the systemic causes of a maintenance problem of any of our system/application like SLR and B2BC.

A structured and logical approach is required for an effective root cause analysis. So, we can follow a series of steps for RCA-

Figure 107. Steps for RCA

Form a RCA team: We need to make a team in between maintenance those who will collect all the data from support team and analysis whether the issue came from ML Model side or application side or server side.

Define the problem: Besides issue analysis, RCA team will list down all the issues and define them in detailed way.

Identify root cause: After defining all the issues, team will analyze the root cause of those issue for which the user or system or DL Model is facing problem.

Root cause corrective action: In this phase, presence of team lead is necessary. When root cause has been found out. It need to be fix and fix in a way by which the issue never happens in future. After fixing the solution need to be implement.

Root cause preventive action: After implementing the solution dev team and RCA team need to be sit down and implement a strategy through which it can be assure that the issue will never happen again.

Service Level Agreement: The Service Level Agreement (SLA) for all SD 17 components, SLR applications including T2SP and B2BC sets forth the commitments provided by the **Genweb2**.

Genweb2 is committed to provide a satisfactory level of support for the whole life cycle of customer services. **Genweb2 Issue Support Line** will be always ready to respond non-critical and critical issues that the user might be experiencing with the service. Considering the severity and time of reporting the SR, the Mean Time to Attend (MTTA) and Mean Time to Resolve (MTTR) for the service is given below –

Priority Code	Definition	MTTA (Mins)	MTTR (Hrs.)
P1	Critical/Major	30	8
P2	Moderate	60	15
P3	Minor	120	48

Table 16. Mean Time to Attend (MTTA) and Mean Time to Resolve (MTTR)

Notes on Priority Definition's:

Priority	Descriptions
Critical/Major	It's an issue for which model gets impact and services became offline. It may be the security issue also.
Moderate	May be the functional or performance issue which doesn't affect directly to the ML model or core services like SLR
Minor	The issue is not related with DL Model or services.

Table 17. Priority Definition

Project Team with RACI Matrix: The RACI matrix is a responsibility assignment chart that maps out every task, milestone or key decision involved in completing a project and assigns which roles are Responsible for each action item, which personnel are Accountable, and, where appropriate, who needs to be Consulted or Informed.

The four roles that our Genweb2 team might play in SD-17 project include the following:

- Responsible (R)
- Accountable (A)
- Consulted (C)
- Informed (I)

	Team Lead/ Project Manager	Machine Learning Expert	Software Engineer/Designer	Tester	Document Analyst
Document Analysis	A/R	C	I	I	R
System Design	A/C	C	C	I	C
DL Model Training	C	R	C	I	I
Application Development	C	C	R	A	I
Application Design	C	I	C	C	C

Table 18. RACI Matrix

Logging & Notifying: Genweb2 will form a dedicated maintenance team those who will keep all the log properly for each and every changes that will happen with any of SD-17 applications. For every changes in SLR, T2SP and B2BC, it needs to get approval from designated approving authority, team lead of maintenance team. On the other hand, if there is any changes like updating version than all the registered user will get an automatic notification.

11. Abbreviations and Acronyms

Term	Abbreviations
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
CV	Computer Vision
SLR	Sign Language Recognition
T2SP	Text to Sign Puppet
B2BC	Bangla to Braille Conversion
UI/UX	User Interface/User Experience
TOR	Terms of Reference
API	Application Programming Interface
OS	Operating System
SLA	Service-Level Agreement
ML	Machine Learning
AI	Artificial Intelligence
DL	Deep Learning
RCA	Root Cause Analysis
SR	Service Request
MTTA	Mean Time to Attend
MTTR	Mean Time to Resolve
NLTK	Natural Language Toolkit
NER	Named Entity Recognition
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
HMMs	Hidden Markov models
SVM	Support Vector Machine
CoNLL	Conference on Natural Language Learning
CRF	Conditional Random Field
AR	Augmented Reality
ISWA	International SignWriting Alphabet
SW	SignWriting
SDLC	Software Development Life Cycle
RMSE	Root Mean Squared Error
CI	Continuous Integration
CD	Continuous Delivery
MLLC	Machine Learning Life Cycle
MLOps	Machine Learning Operations
GAN	Generative Adversarial Network
VGG	Visual Geometry Group
LSTM	Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers

12. References

[1] H. Academy of Sciences and U. o. Hamburg, "DGS-KORPUS," [Online]. Available: https://www.sign-lang.uni-hamburg.de/dgs-korpus/index.php/. [Accessed 15 01 2022].

[2] Hiventures, "SignAll," Hiventures, [Online]. Available: https://www.signall.us/. [Accessed 16 01 2022].

[3] E. Mohamed and B. Catherine, "KinTrans-Hands Can Talk," KinTrans Inc., [Online]. Available: https://www.kintrans.com/. [Accessed 15 01 2022].

[4] G. Wu, "Kinect Sign Language Translator," Microsoft Research, [Online]. Available: https://www.microsoft.com/en-us/research/blog/kinect-sign-language-translator-part-1/. [Accessed 17 01 2022].

[5] B. Danielle, K. Oscar, B. Mary, B. Larwan, B. Patrick, B. Annelies, C. Naomi and H. Matt, "Sign language recognition, generation, and translation: An interdisciplinary perspective.," 2019.

[6] C. Zhe, H. Gines, S. Tomas, S. Yaser and W. Shih-En, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *IEEE Transactions on Pattern Analysis & Machine Intelligence,* vol. 43, no. 1939-3539, pp. 172-186, 2018.

[7] C. C. Necati, H. Simon, K. Oscar, N. Hermann and B. Richard, "Neural Sign Language Translation," pp. 7784-7793, 2018.

[8] G. G. Binyam, W. Peter and H. Tom, "Automatic sign language identification," *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings,* pp. 2626-2630, 09 2013.

[9] K. Dimitrios, D. Kosmas and D. Petros, "Sign Language Recognition based on Hand and Body Skeletal Data," *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON),* 05 2018.

[10] Readthedocs, "Read the Docs, Inc & contributors," Readthedocs, [Online]. Available: https://detectron2.readthedocs.io/en/latest/tutorials/training.html. [Accessed 16 01 2022].

[11] G. Scholar, "Google Scholar," Google, [Online]. Available: https://scholar.google.com/scholar?q=Faster+R-CNN+FPN+architecture&hl=en&as_sdt=0&as_vis=1&oi=scholart. [Accessed 16 01 2022].

[12] Vkhalidov, R. Girshick and A. Guler, "Github," 14 01 2019. [Online]. Available: https://github.com/facebookresearch/DensePose/blob/main/GETTING_STARTED.md. [Accessed 16 01 2022].

[13] Google, "MediaPipe," Google LLC., [Online]. Available: https://google.github.io/mediapipe/solutions/face_mesh.html. [Accessed 16 01 2022].

[14] G. LLC, "MediaPipe," Google LLC, [Online]. Available: https://google.github.io/mediapipe/solutions/hands.html. [Accessed 16 01 2022].

[15] O. and H. , "Colah's blog," Github.io, [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 17 01 2022].

[16] J. Kunal, R. Sunil and S. Simran, "Analytics Vidhya," Analyticsvidhya.com, [Online]. Available: https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/. [Accessed 17 01 2022].

[17] S. Platform, "Cornell University," arXiv, [Online]. Available: https://arxiv.org/abs/1810.04805. [Accessed 17 01 2022].

[18] S. Benoit, A. Julien, R. Laurent and S. O. Pedro, "Oscar," Inria, [Online]. Available: https://oscar-corpus.com/. [Accessed 17 01 2022].

[19] Wikimedia, "bnwiki," Wikimedia, [Online]. Available: https://dumps.wikimedia.org/bnwiki/latest/. [Accessed 17 01 2022].

[20] J. Devlin, "Google-research/bert," Github, [Online]. Available: https://github.com/google-research/bert. [Accessed 17 01 2022].

[21] L. and R. , "Datadets:wikiann," Hugging Face, [Online]. Available: https://huggingface.co/datasets/wikiann. [Accessed 17 01 2022].

[22] f. hugging, "bert-base-multilingual-uncased," Hugging Face, [Online]. Available: https://huggingface.co/bert-base-multilingual-uncased. [Accessed 17 01 2022].

[23] f. hugging, "xlm-roberta-base," Hugging Face, [Online]. Available: https://huggingface.co/xlm-roberta-base. [Accessed 17 01 2022].

[24] K. Anoop, K. Mitesh and K. Pratyush, "ai4bharat/indic-bert," Hugging Face, [Online]. Available: https://huggingface.co/ai4bharat/indic-bert. [Accessed 17 01 2022].

[25] H. Matthew and M. Ines, "spacy.io/api," MIT, 02 2015. [Online]. Available: https://spacy.io/api. [Accessed 16 01 2022].

[26] H. Matthew and M. Ines, "spacy.io/usage/models," spacy.io, 02 2015. [Online]. Available: https://spacy.io/usage/models. [Accessed 16 01 2022].

[27] B. Georg, T. Sphinx and A. Tom, "NLTK," Sphinx & NLTK Theme, [Online]. Available: https://www.nltk.org/. [Accessed 16 01 2022].