Chapter 1
INTRODUCTION

1.1 Background

The World Health Organization (WHO) [8] reports that there were 278 million hearing or listening impairment cases in 2005 and 466 million cases in early 2018. By 2050, it is predicted that this number would rise to 400 million [8]. This deaf population communicates by a system of signs known as sign language, which varies depending on the country. To put it another way, sign language (SL) is a nonverbal communication language that makes use of visual sign patterns formed with the hands or other body parts and is typically utilized by people who have hearing loss or other hearing impairments. Full-fledged natural languages and sign languages (SLs) have their own lexicon and syntax. For deaf communities, a variety of SLs including Nepali Sign Language(NSL) American Sign Language (ASL), Australian Sign Language (ASL), British Sign Language (BSL), Danish Sign Language (DSL), French Sign Language (FSL), and many others have been established. The SLs are not mutually comprehensible and universal, despite some apparent similarities. Even while both ASL and BSL employ the same verbal language, they are different. Even the national sign language is quite challenging for the average hearing and listening person to grasp. Thus, qualified SL interpreters are required for appointments with doctors and lawyers, for training and educational purposes, etc. An effective communication interface between those who have hearing or listening impairments and sighted people can be created by the automatic recognition of an SL and its translation into natural language.

Very few works have been performed regarding the interpretation of Nepali Sign Language(NSL) despite it being used as the primary source of communication by the Deaf and Dumb(D&D) here in Nepal. The National Federation of the Deaf Nepal (NDFN) has developed and published a set Nepali Sign language Dictionary consisting of all 2 forms of signs as proposed by [3]. The alphabets published by NDFN are:

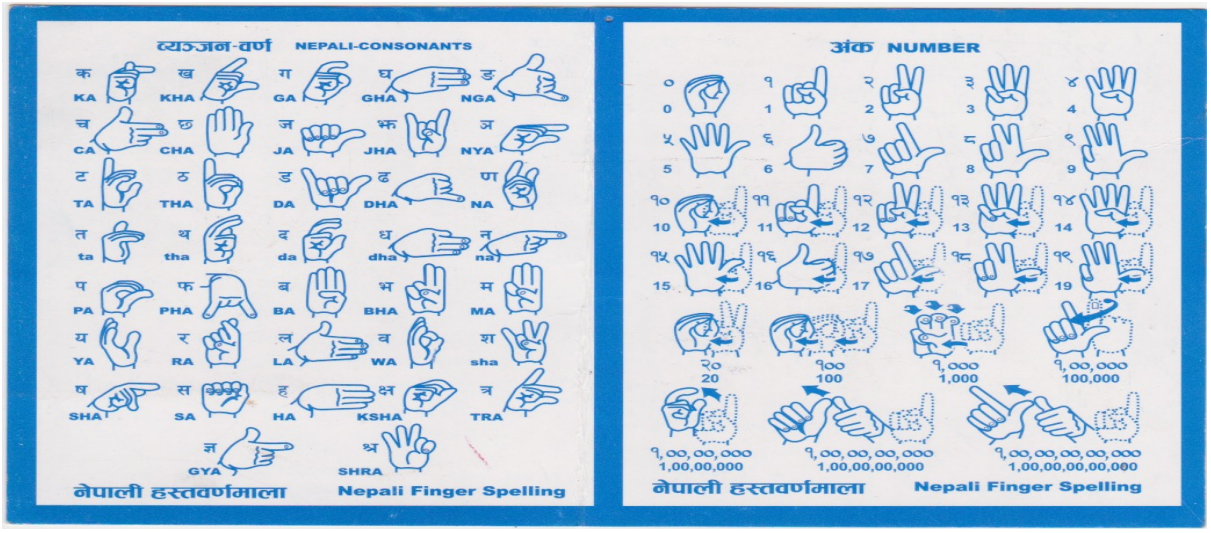

Fig: 1.1 NSL of Nepali Devnagari Consonants and Numbers

1.2 Statement of Problem

- There is no reliable and ample dataset for Nepali Sign Language.
- Very little research and development have been done to interpret NSL.
- Most of the work done is focused on Static Signs Based Recognitions though almost all communication is done by the D&D in Continuous Sign Sentence Recognition.
- There is no easy and smooth communication medium between the D&D and normal communicators.

1.3 Motivation

The very reason that there is a lack of an easy and applicable communication platform between the D&D and normal communicators in NSL is our biggest source of motivation for initiating FISPAN. Aside from that, creating a dataset so that NSL interpretation projects can be eased and inspiring for the upcoming students and researchers, was also our encouraging factor.

1.4 Objectives

- Initiating the collection of ample and reliable datasets for Nepali Sign Language.
- Focusing on both static and dynamic Sign recognition.

1.5 Scope and Application

This project can be used by D&D people to communicate with normal people that detects basic gestures, Nepali alphabets and numbers.

Chapter 2
LITERATURE REVIEW

Designed for the Deaf and Dumb (D&D), sign language is a gesture-based language that includes hand movements, positioning of hands, and facial expression for communication [1]. This type of language is not universal and varies among many nations. A few could be named as Nepali Sign Language(NSL), Indian Sign Language(ISL), American Sign Language(ASL) etc.According to [2], the Sign Language Recognition(SLR) system can be broadly classified into 3 types:

1. Fingerspelling Recognition
2. Isolated Word Recognition
3. Continuous Sign Sentence Recognition.

However, based on the type of signs, [3] classified SLR into two types :

1. Static Signs-based Recognitions
2. Dynamic Signs-based |Recognitions.

More needs to be done in detecting and translating Nepali Sign Language translation when we compare the research and outputs of ASL and other variations. Computer vision-based and sensor-based systems are two categories in which the recognition models fall. The camera is utilized for input in CV-based gesture recognition, and image processing of the input motions is done before recognition. This processing and training are then done using several algorithms, such as Hidden Markov Model and Neural Network Techniques. The fundamental issue with a vision-based sign language identification system is that there are numerous environmental uncertainties involved in the image acquisition process, such as where the camera is located, the lighting situation, and backdrop variables. [1]

[4] talked about how hand motions for human-computer interaction could be seen visually. The study, which was published in 1997, focuses on the benefits and drawbacks of using either a 3D model of the human hand or an image appearance model of the human hand for interpreting gestures. When this study was conducted, 3D hand models offered a method to depict hand movements, but they also presented computational challenges that had not yet been overcome given the need for HCI(Human Computer Interaction) to operate in real-time. They also talked about existing gestural systems and other potential uses for gesture recognition using vision.

A computer vision-based hand gesture recognition using a Convolutional Neural Network on python was proposed by [1]. Using Python libraries and programs like OpenCV and skimage, the gesture datasets were preprocessed. The CNN VGG-16 model was then used to train the datasets. The recognized input was spoken after conversion. As a person who does not understand sign language will be shown the meaning of the hand signs, this allowed for

one-way communication. This study also introduced text-to-sign language conversion, which enables a person who does not understand sign language to convert text to sign language fingerspelling that the signer would understand, enabling two-way communication.

Hidden Markov Models (HMMs) for a real-time system designed to recognize continuous Chinese Sign Language were provided with the data from [5] as input (CSL). A 3-D tracker, two Cyber-Gloves, and raw data were collected. The training sentence was divided into basic units using the dynamic programming (DP) technique, and the Welch-Baum algorithm was utilized to estimate. Results of tests utilizing 220 words and 80 sentences revealed 94.7% recognition rates for the system.

Parallel Hidden Markov models (PaHMMs) were utilized by [6] to recognize American Sign Language. They claimed that for a continuous recognition system, phonemes may be employed in place of complete signs. Used If any word can be divided into basic phonemes in the same way that words are in speech recognition, then there should be two channels for the right and left hands. A modest vocabulary number (22 signs) was tested using a single channel of the HMM model, and the results showed an accuracy rate of 87.88%. A more extensive vocabulary, hand arrangement, orientation, and facial emotions cannot be recognized by the system.

A sensor-based real-time hand gesture recognition system for ISL translation was introduced by [7] in 2019. The data from the sensor has been analyzed in this paper to extract hand orientation and finger movements, which have then been wirelessly transmitted to the processing device. Finally, the classification is done using an LSTM network. 26 frequently used ISL gestures are used to test this model.

Chapter 3

FEASIBILITY STUDY

3.1 Overview

A project's key factors, such as its economic, technical, legal, and scheduling considerations, are examined in a feasibility study with the aim of successfully completing the project. Before investing time and money in a project, it weighs its benefits and drawbacks. It establishes whether the proposed system conflicts with the requirements or not. Operationally, for instance, spacing, timing, software legality, and so forth. A report that thoroughly assesses the frameworks of analysis for a certain project is known as a project feasibility study.

3.1.1 Time Feasibility

The projected development period for this project is 5 months, possibly longer. In order to make the project more useful while still reaching the deadline, we want to conclude it as soon as reasonably practicable.

3.1.2 Economic Feasibility

The vast majority of online tools, libraries, frameworks, and data sets should be open-source. There is therefore no cost. No financial resources have been allocated to this project. According to the project's line of code (LOC), perhaps some investments will be needed in the future.

3.1.3 Operational Feasibility

We are developing a Sign Language Translation System for Nepali Sign Language. Multiple works have already been done in other standards. Though the language differences, overall systems can be considered for references.

Chapter 4

PROJECT MANAGEMENT AND OVERVIEW

4.1 Agile Development

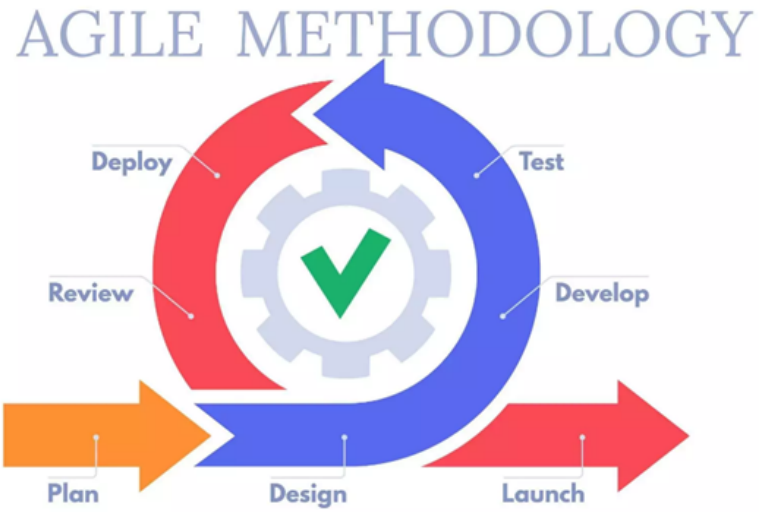

Fig: 4.1.1 Agile Development Cycle

Agile development refers to a development approach based on iterative , incremental, evolutionary development. Agile methods break tasks into smaller iterations, parts or sprints that minimize the amount of up-front planning and design.

Our project can be broadly divided into Data Accquisition, Pre-Processing, Classification and Prediction. All these four phases can be developed independently till a certain level. Since Agile allows us to work on each phases iteratively, it helps us to save a lot of time with parallel progress. Tht is the reason we have chosen Agile Management to execute our project.

4.2 Gantt Chart

Fig 4.2.1 Gantt Chart

4.3 General Usage Flowchart

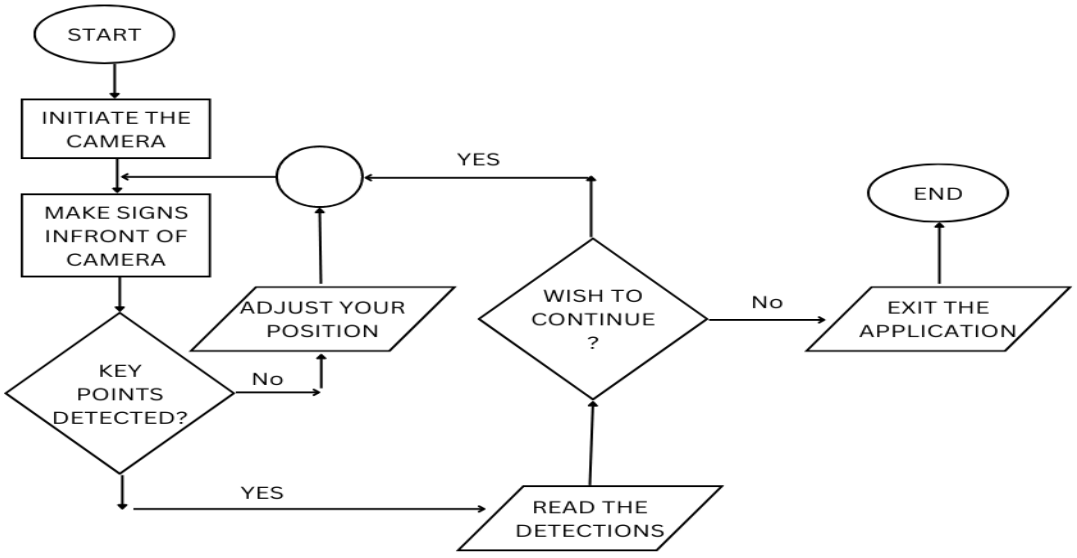

Fig 4.3.1 General Usage Flowchart

4.4 Use- Case Diagram

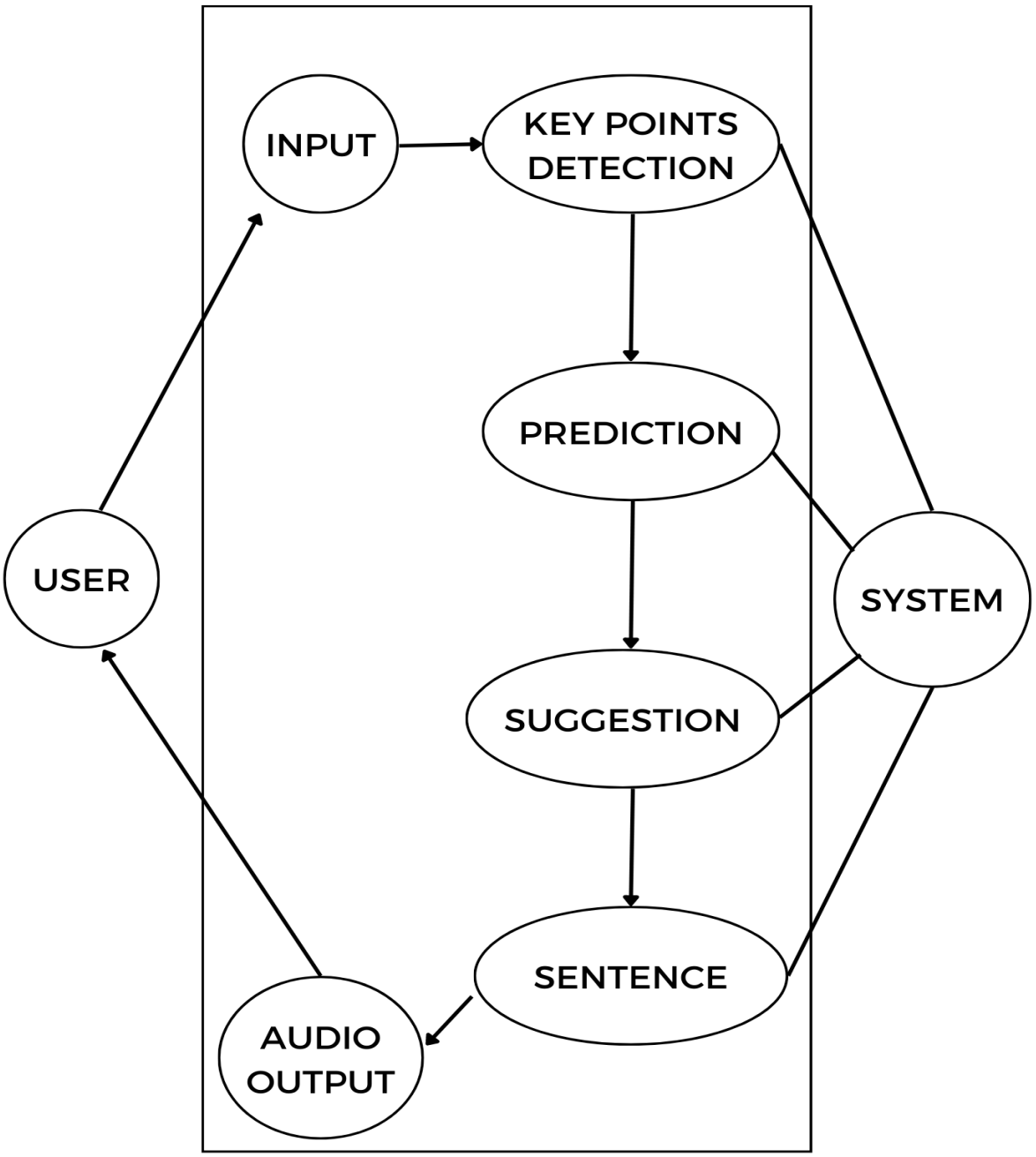

Fig 4.4.1 Use-Case Diagram

4.5 ER Diagram

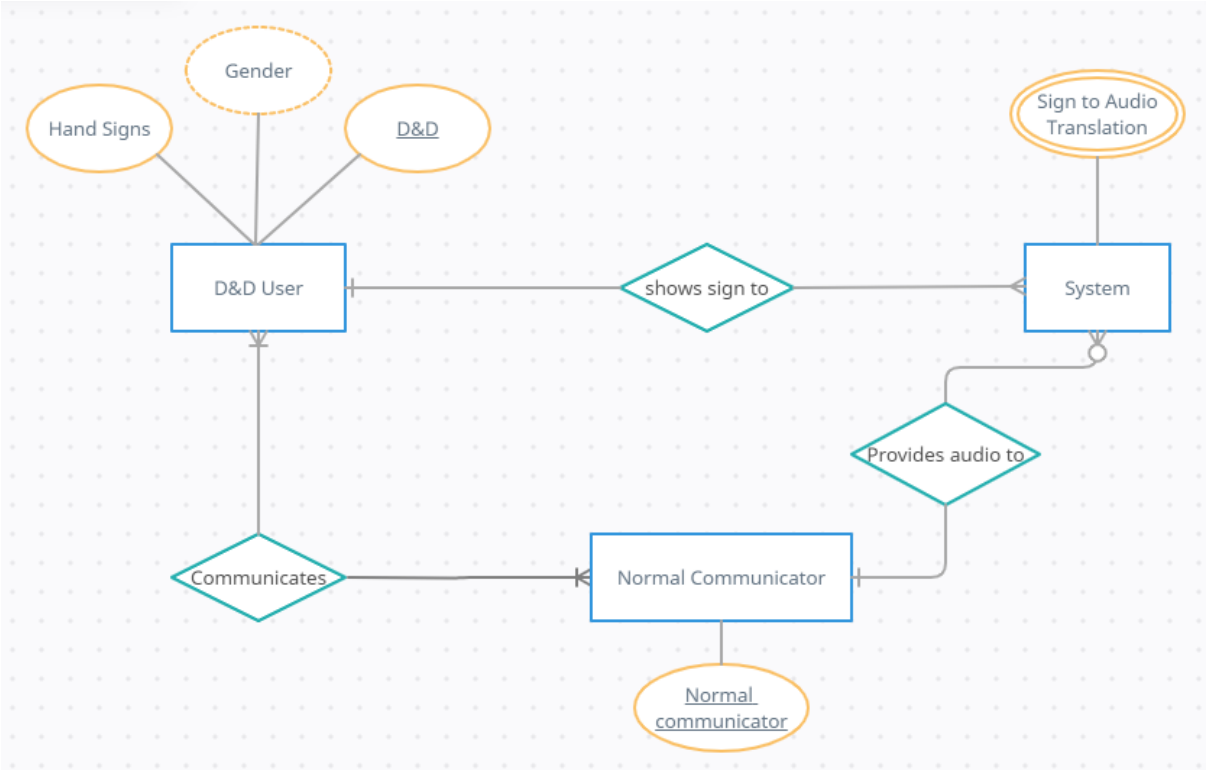

Fig 4.5.1 ER Diagram

4.6 Context Diagram

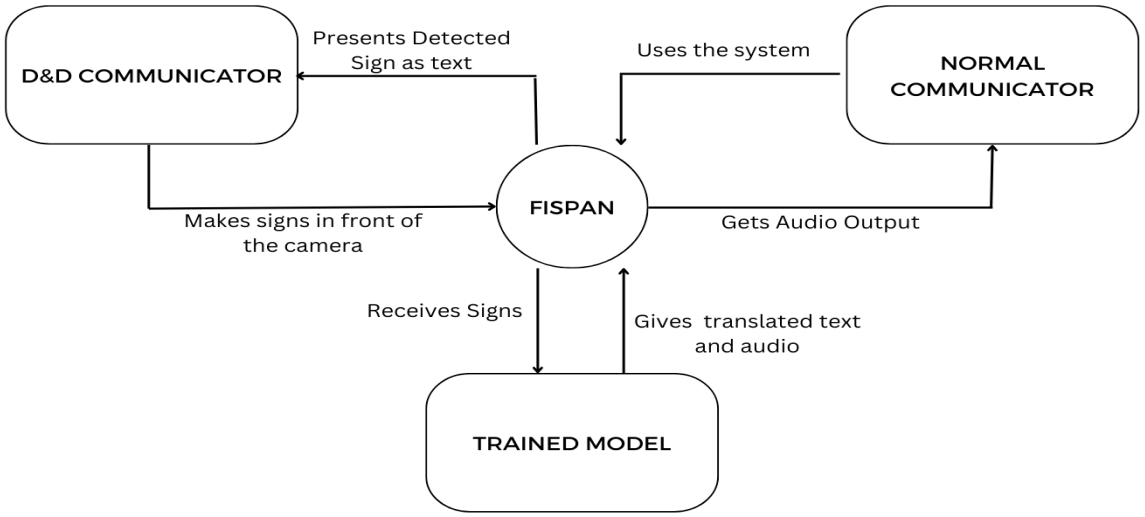

Fig 4.6.1 Context Diagram

4.7 Documentation and Referencing

The documentation process has been going parallel to the development phases. The satndatrd format provided by the Department of Computer Engineering, Khwopa Engineering College has been used. The IEEE method of referencing and citation has been used to cite references.

Chapter 5

TOOLS AND TECHNOLOGIES

5.1 Overview

For executing this project, we have installed and used many tools, libraries and concepts. Following list contains list of what we have used so far and what we plan to use.

5.2 Machine Learning and Deep Learning

Machine learning is the study of how to make computers behave without being explicitly programmed. Machine learning has enabled self-driving cars, accurate speech recognition, successful web search, and a deep comprehension of the human genome. Machine learning underpins everything, including driverless vehicles and translation software. It provides a mechanism to resolve issues and respond to challenging queries. In essence, it is a process of teaching an algorithm or model—a type of computer program—to reliably anticipate outcomes from data. There are 4 types of machine learning strategies:

a. Unsupervised Learning
b. Supervised Learning
c. Semi-Supervised Learning
d. Reinforcement Learning

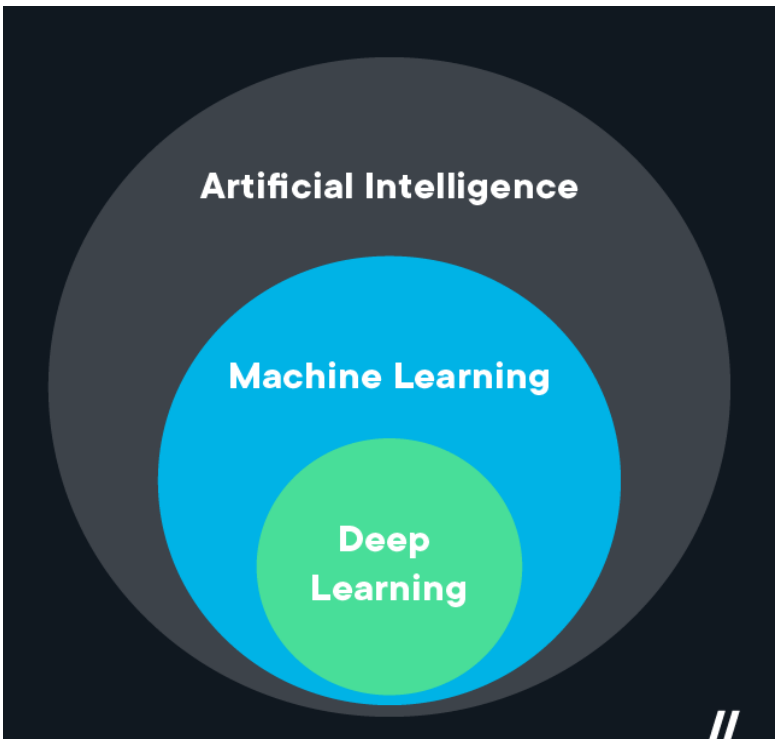

Fig 5.2.1 Relationship between AI, Machine Learning and Deep Learning

Deep learning is a machine learning method that instructs computers to learn by doing what comes naturally to people. Driverless cars use deep learning as a vital technology to recognize stop signs and tell a pedestrian from a lamppost apart. It is essential for voice control on consumer electronics including hands-free speakers, tablets, TVs, and smartphones. Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. [9]

5.3 Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

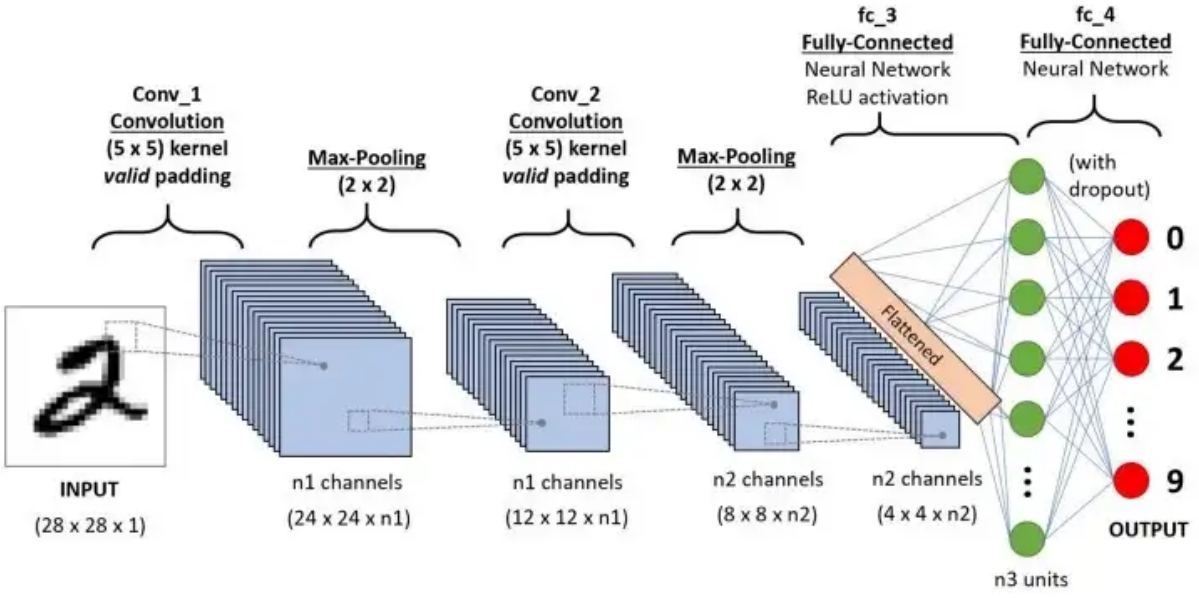

Fig 5.3.1 Convolutional Neural Netwrk and its layers

5.4 Recurrent Neural Networks

An artificial neural network that employs sequential data or time series data is known as a recurrent neural network (RNN). These deep learning algorithms are included into well-known programs like Siri, voice search, and Google Translate. They are frequently employed for ordinal or temporal issues, such as language translation, natural language processing (nlp), speech recognition, and image captioning. Recurrent neural networks (RNNs) use training data to learn, just like feedforward and convolutional neural networks (CNNs) do. They stand out due to their "memory," which allows them to affect the current input and output by using data from previous inputs. Recurrent neural networks' outputs are dependent on the previous parts in the sequence, unlike typical deep neural networks, which presume that inputs and outputs are independent of one another. [10]

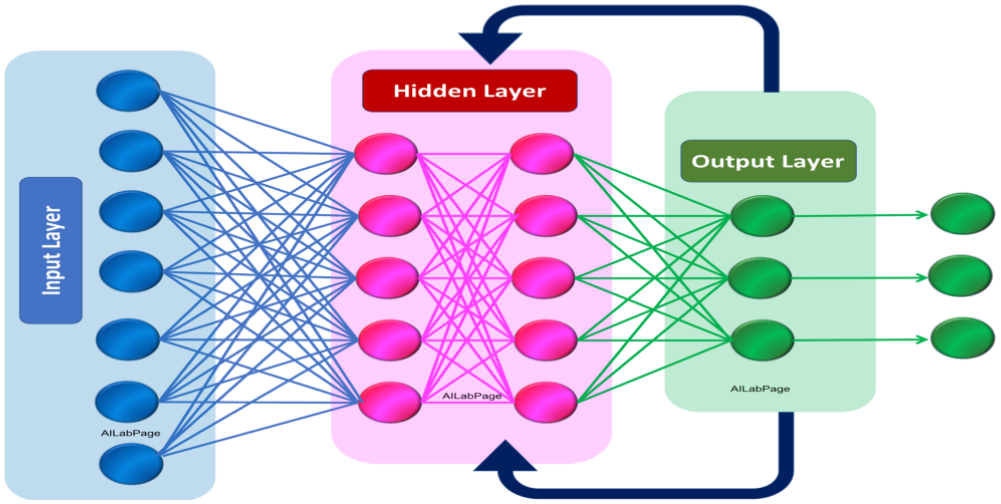

Fig 5.4.1 Recurrent Neural Network

5.5 LSTM

Long Short Term Memory Networks, most commonly referred to as "LSTMs," are a unique class of RNN that can recognize long-term dependencies. They were first presented by Hochreiter & Schmidhuber (1997), and numerous authors developed and popularized them in subsequent works. 1 They are currently frequently used and perform incredibly well when applied to a wide range of issues.

Intentionally, LSTMs are created to prevent the long-term reliance issue. They don't struggle to learn; rather, remembering information for extended periods of time is basically their default behavior. All recurrent neural networks have the shape of a series of neural network modules that repeat. 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.

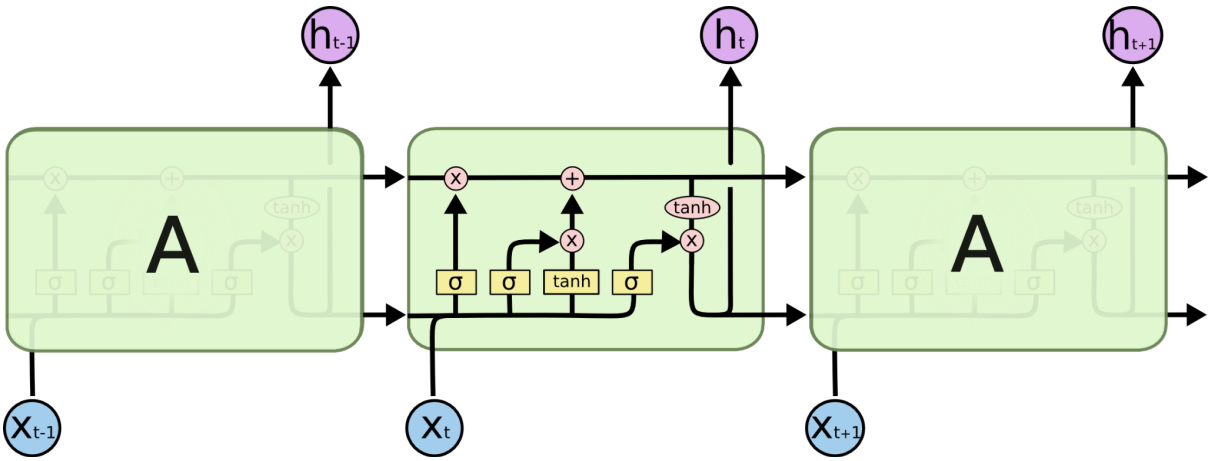

Fig 5.5.1 LSTM

The main reason to use LSTM are:

- Less Data required
- Faster to train
- Faster Detections

5.6 Tensorflow

A free and open-source machine learning and artificial intelligence software library is called TensorFlow. Although it has many uses, its main emphasis is on deep neural network training and training and inference. Python, JavaScript, C++, and Java are just a few of the many programming languages that TensorFlow is compatible with. This versatility supports a broad range of applications across numerous sectors. We have used version 2.5.1

5.7 Tensorboard

TensorBoard provides the visualization and tooling needed for machine learning experimentation:

- Tracking and visualizing metrics such as loss and accuracy
- Visualizing the model graph (ops and layers)
- Viewing histograms of weights, biases, or other tensors as they change over time
- Projecting embeddings to a lower dimensional space
- Displaying images, text, and audio data
- Profiling TensorFlow programs

5.8 Keras

Keras is a python-based deep learning framework that is open source. Francois Chollet, a Google artificial intelligence researcher, came up with the idea. Keras adheres to best practices for lowering cognitive load, such as providing uniform and simple APIs, limiting the number of user activities required for typical use cases, and providing clear and actionable error messages. It comes with a lot of documentation and developer instructions. Keras is presently used by Google, Square, Netflix, Huawei, and Uber, among others.

5.9 Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant.

5.10 Open CV

OpenCV (Open-Source Computer Vision) is a programming library for real-time computer vision that is open-source. Image processing, video recording, and analysis for features such as face and object identification are the most common applications. It is written in C++ and uses it as its primary interface, although it also has Python, Java, and MATLAB/OCTAVE connections.

5.11 Adam Optimizer

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. [12]

5.12 Python

Python is an interpreted high-level general-purpose programming language. Its design emphasizes code readability by using a lot of indentation. The language features and object-oriented approach of python are designed to help programmers write clear, logical code for both small and large-scale projects

5.13 Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Chapter 6

METHODOLOGY

6.1 Overview

Solving a problem on a computer requires a thorough analysis of the potential data. Once the problem has been analyzed, a detailed procedural solution can be developed. The program or project development process has been planned using the Agile Development Method. Also, it uses a vision-based methodology. The issue of employing any artificial gadgets for interaction is eliminated because all signs are represented with bare hands. We have divided the whole project methodology into three parts:

a. Image Capturing and Pre-Processing
b. Classification
c. Prediction

6.2 Image Capturing and preprocessing

The images for the first five nepali Devnagari letters "क", " ख", "ग", "घ", and "ङ" were collected manually. For that, hand-signs of about 1000 people were collected for each letters. Then the each images were sorted and stored in individual folders with theri respective names.

However, for capturing some dynamic signs, we also setup our webcams and by using the "Video capture and split frame" technique, we collected 30 frames worth of data for 'क', 'Sunday', 'Red', 'Father' and '1', and they all have been stored as a numpy array in their respective folders.

After we collected all the required data, we have converted the manually collected data into Gray Scale Images for now. However, for the dynamic signs, we captured the video frames by using mediapipe which helped us to detect landmarks in our face, right hand, left hand and shoulders. We now have 30 videos worth of data with each video of 30 frames in length. And we are using the web-cam collected data for now to present in this report.

For labelling, we created X_data and y_data that represented all the images in the dataset and their respective labels. Then, we did a train-test split, and obtained (X_train,y_train) and (X_test, test) as our training and testing NumPy array in the ratio of 80:20. Currently the dataset consists of alphabets from "क" to "1" and another alphabets and signs dataset are still being completed. We also generated a category list that had all the category names i.e. folder names such as "क",'Sunday'… '1'. This was used to create a dictionary that would map numbers starting from 0 to (number of categories -1) to the alphabets from "क" to other categories. That is, 0 is mapped to "क", 3 is mapped to '1' and so on if other categories are added.

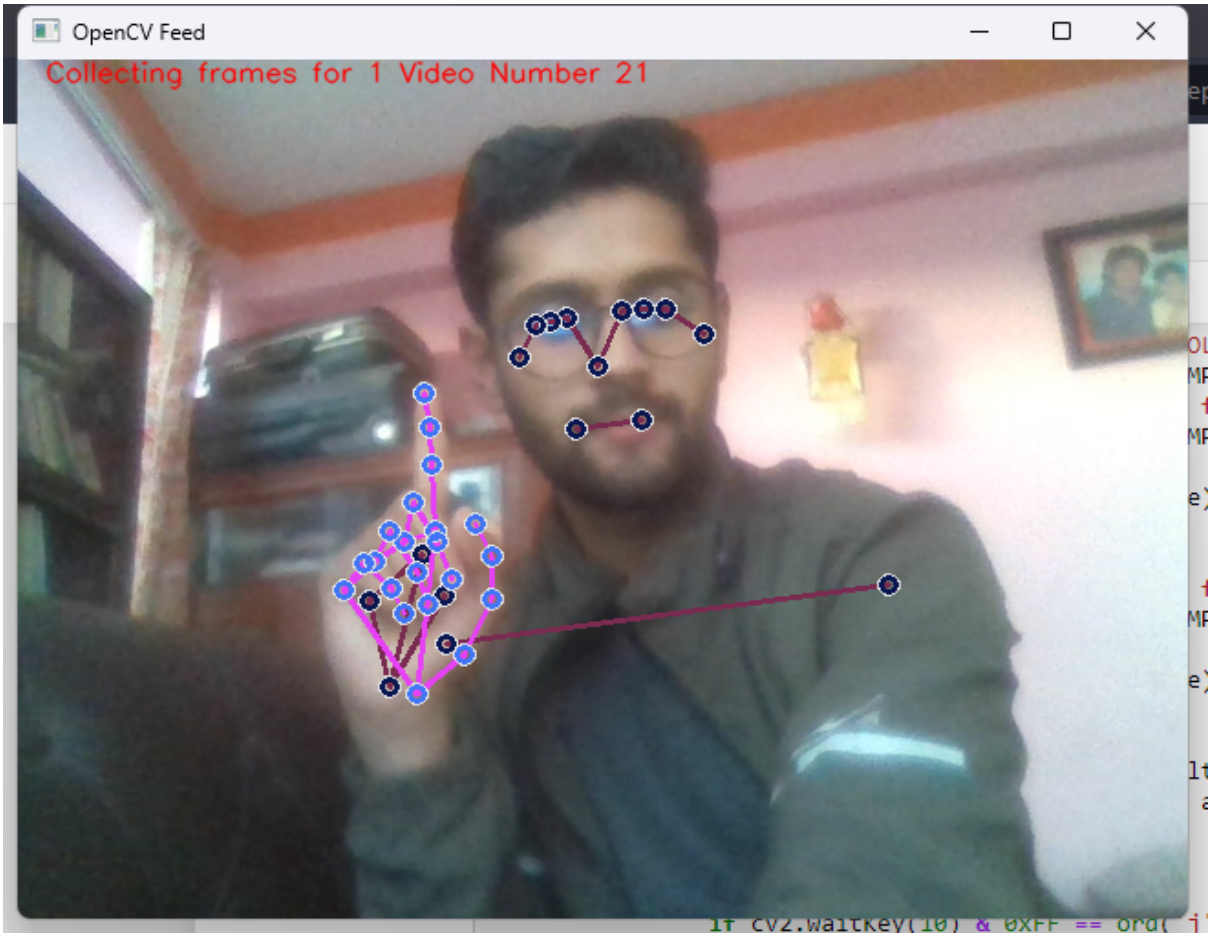

Fig 6.2.1 Data Collection of "1" using webcam

6.3 Classification

We trained the model using LSTM-RNN. We imported the Sequentilas from Keras to build sequential neural networks. We also imported the tensorboard to track the progress of the model in real time. We used the Adam optimizer to update the model in response to the loss functions output. The Adam optimizer combines the benefits of two stochastic gradient descent algorithms: adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp). Adam is a different optimization algorithm that can be used to train deep learning models instead of stochastic gradient descent. Adam creates an optimization technique that can handle sparse gradients on noisy situations by combining the best features of the AdaGrad and RMSProp algorithms.

We used the following algorithm for accurate prediction:

1. After setting up the camera, we collect the landmarks of hands, face, and shoulders and save those as video frames with a split of 30. We save them as numpy array of shape (150, 30, 1662).

2. The processed image is sent into the LSTM model for prediction, and if a letter is detected for more than 30 frames, the letter is printed and used in the word formation.

3. Finally the detected signs are displayed in the UI screen in a row style.

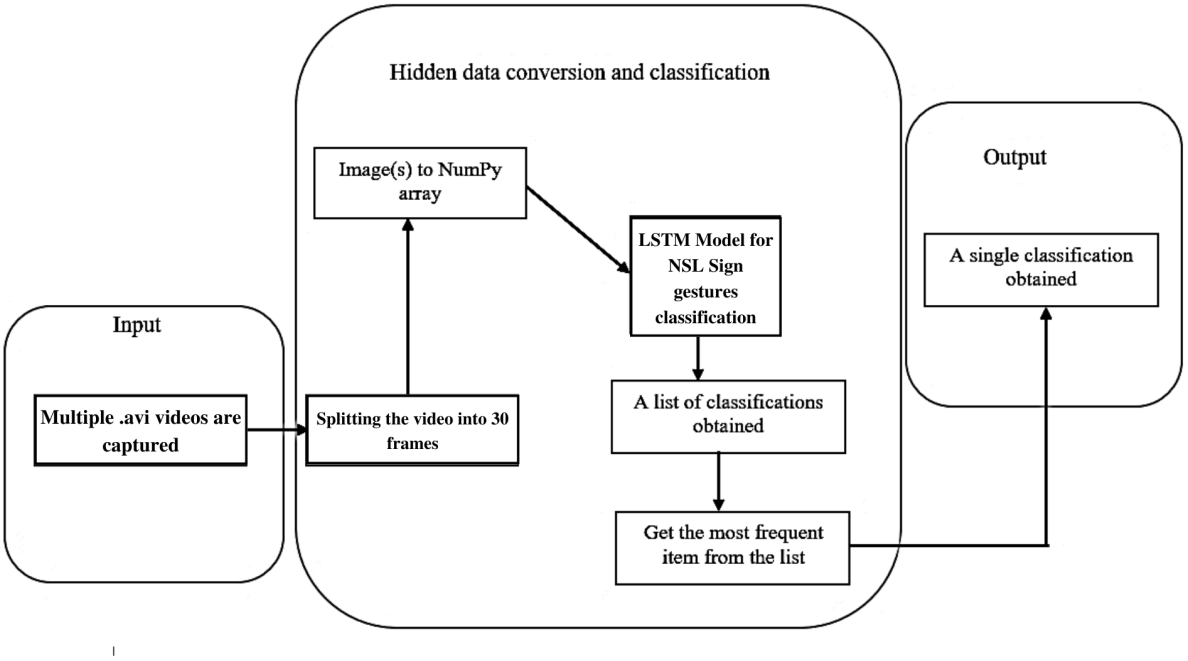

Fig: 6.3.1 Video Capture-split frame and classify Method.

We trained this model using Adam Optimizer with 2000 epochs, we got the validation accuracy of 0.1972 in the first epoch. Then the accuracy increases rapidly to 0.2394 in second epoch and it increases gradually until 2000th epoch. It is because of faster computational time and requires fewer parameters for tuning.

```
model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])

Epoch 1/2000
5/5 [==============================] - 11s 823ms/step - loss: 2.3234 - categorical_accuracy: 0.1972
Epoch 2/2000
5/5 [==============================] - 1s 116ms/step - loss: 5.0821 - categorical_accuracy: 0.1761
Epoch 3/2000
5/5 [==============================] - 1s 110ms/step - loss: 4.0086 - categorical_accuracy: 0.2394
Epoch 4/2000
5/5 [==============================] - 1s 113ms/step - loss: 2.9205 - categorical_accuracy: 0.2254
Epoch 5/2000
5/5 [==============================] - 1s 108ms/step - loss: 3.8699 - categorical_accuracy: 0.2113
Epoch 6/2000
5/5 [==============================] - 1s 110ms/step - loss: 4.9504 - categorical_accuracy: 0.2606
Epoch 7/2000
```

Fig: 6.3.2 Adam Epoch

The following is the total parameters, trainable and non-trainable parameters when the LSTM model was executed.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 64)            442112
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 128)           98816
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 165
=================================================================
Total params: 596,741
Trainable params: 596,741
Non-trainable params: 0
_________________________________________________________________
```

Fig 6.3.3 LSTM Execution

6.4 Prediciton

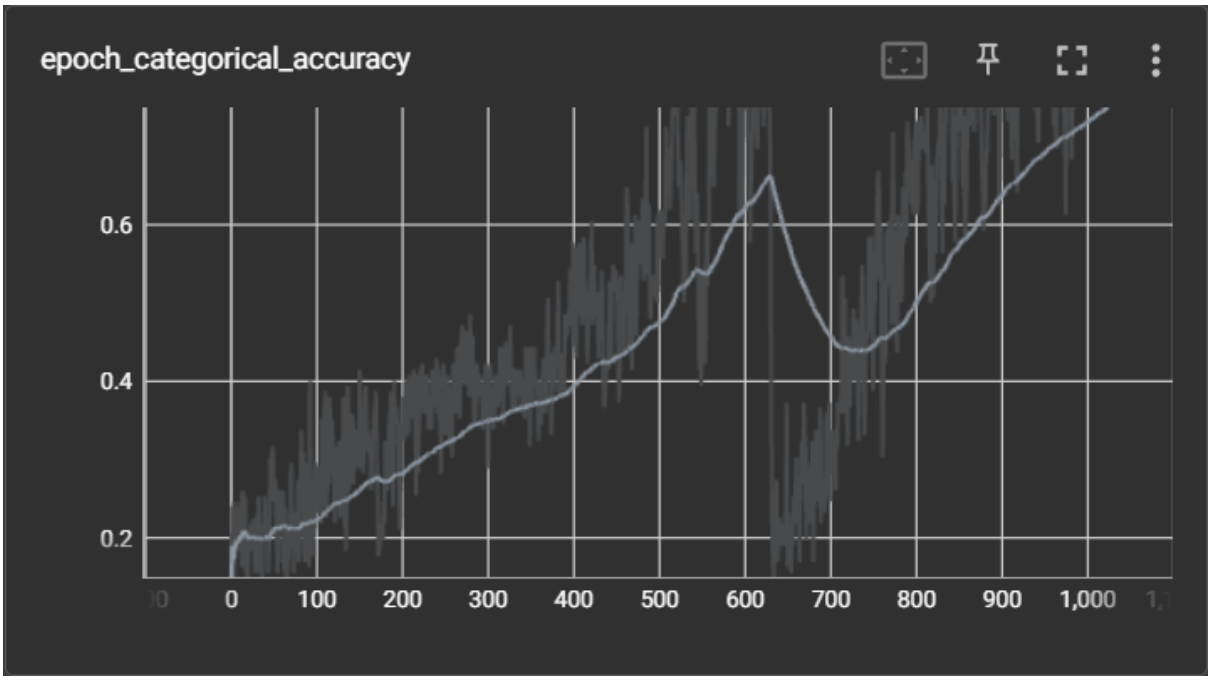

Fig 6.4.1 Epoch Categorical Accuracy

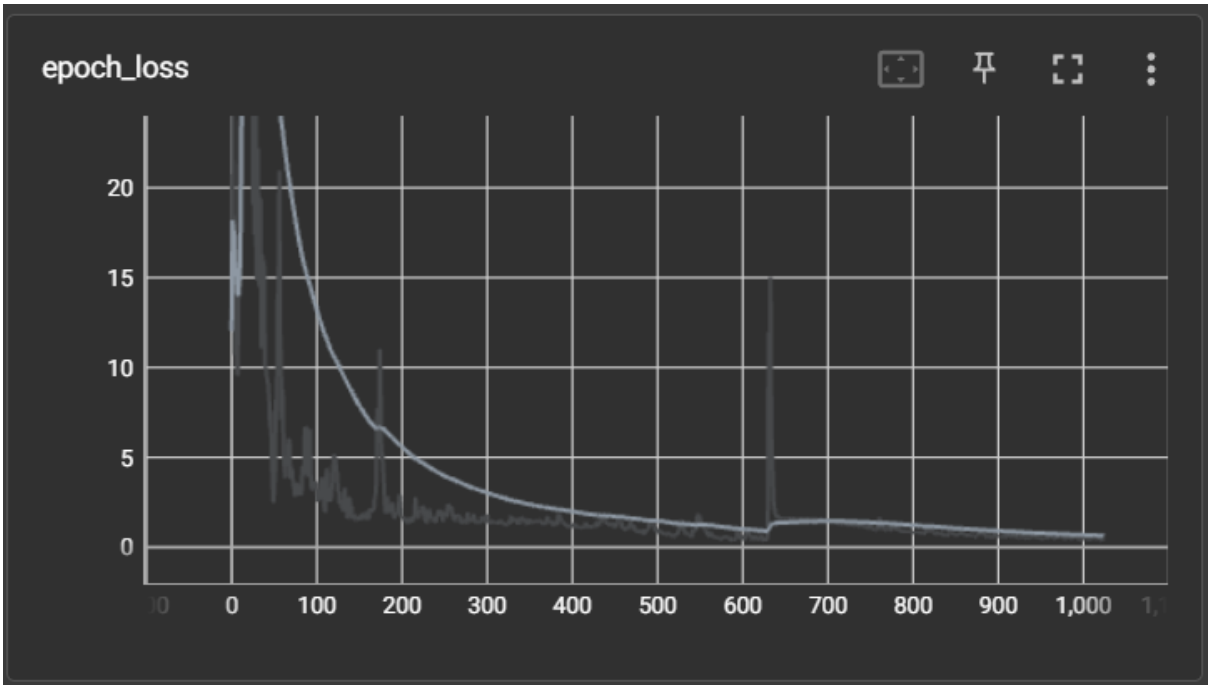

Fig 6.4.2 Epoch Loss

We trained the model for 2000 epoch and got the test data accuracy of 0.66 and training data accuracy of 0.75

```
In [115]: accuracy_score(ytrue, yhat)  #ACCURACY TEST: GOT 75%
Out[115]: 0.75
```

Fig 6.4.3 Train Data Accuracy

```
In [110]: accuracy_score(ytrue, yhat)  #ACCURACY TRAIN: GOT 75%
Out[110]: 0.6619718309859155
```

Fig 6.4.4 Test Data Accuracy.

Chapter 7

RESULTS AND PROGRESS

Overview

By far, our model is able to predict ‘Father’ and ‘Red’ accurately. Following are some screenshots of our predictions.

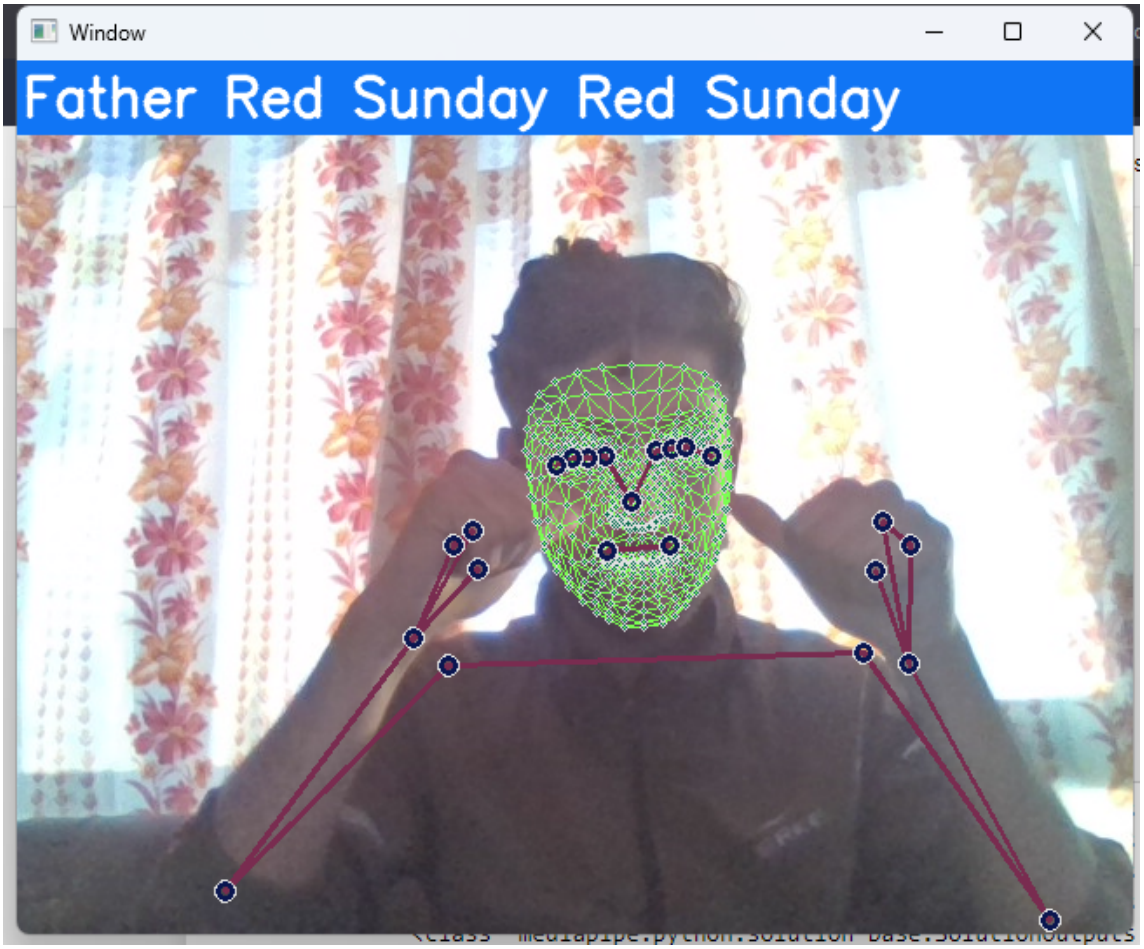

Fig 7.1 Sunday Detection

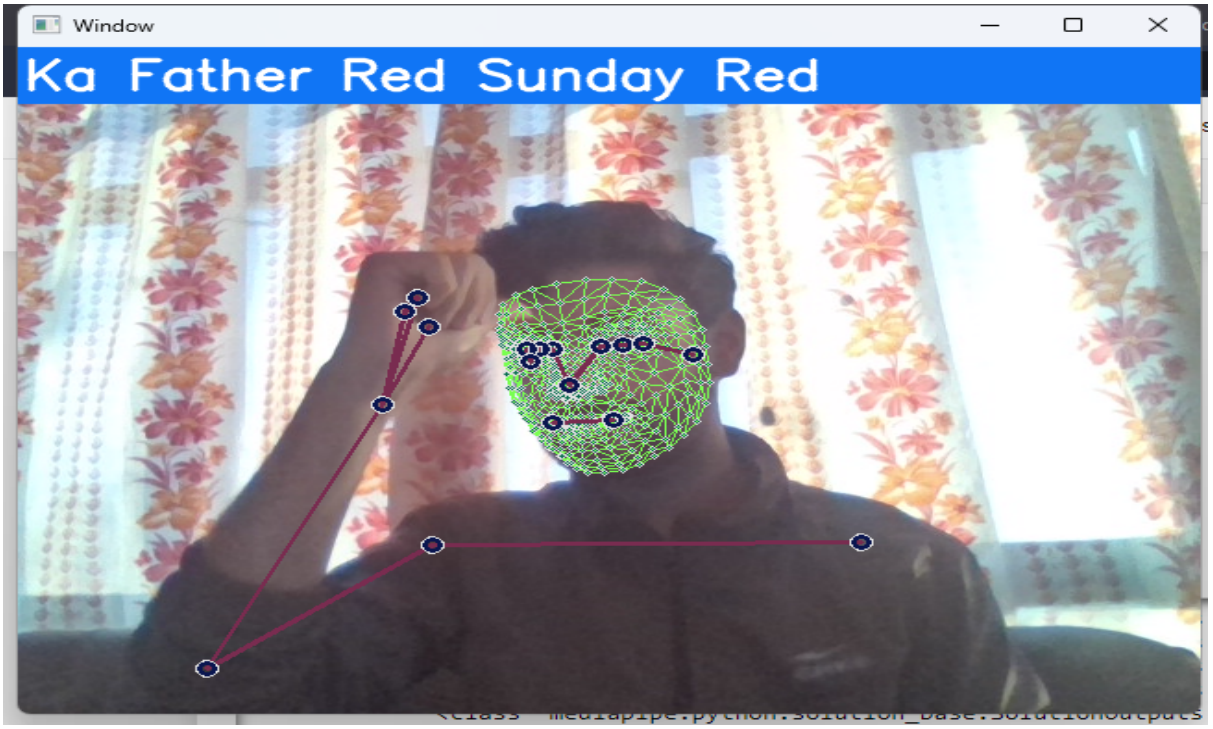

Fig 7.2 Red Detection

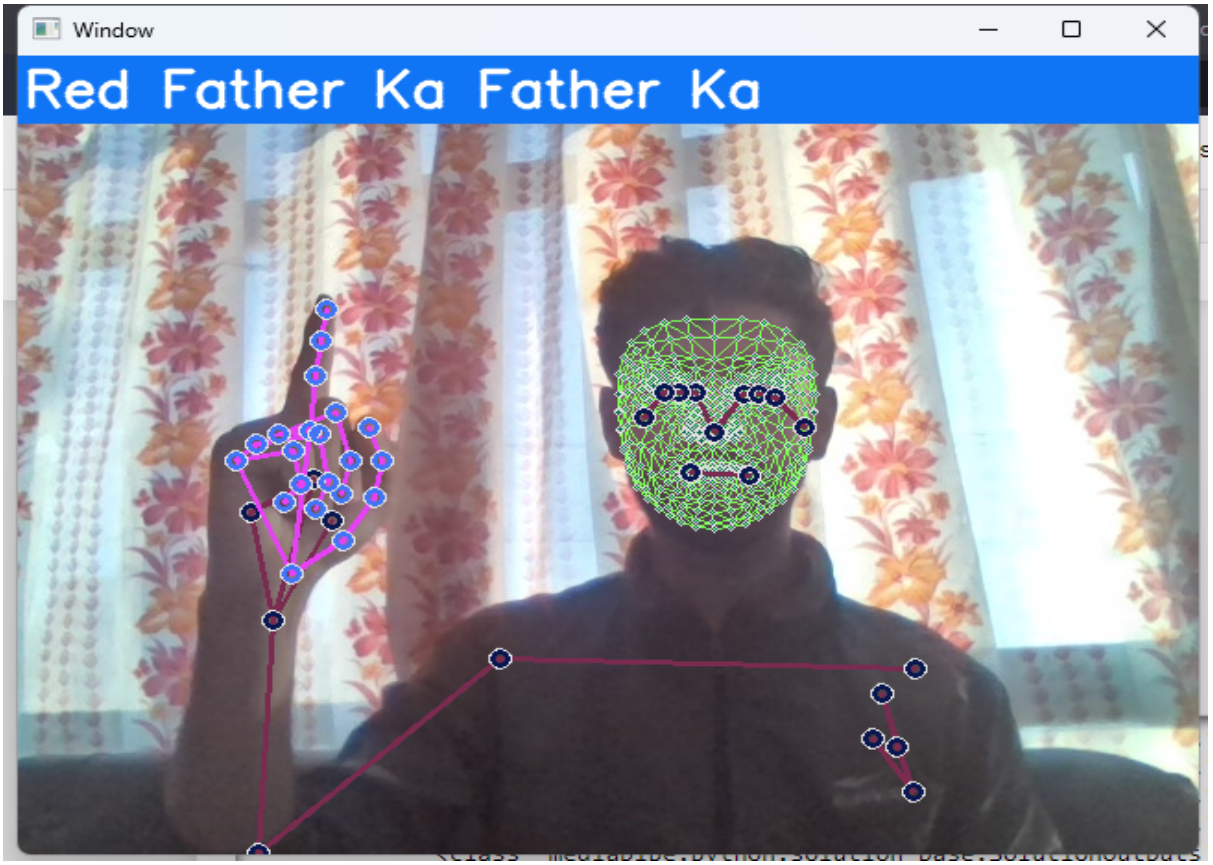

Fig 7.3 Error in detecting ‘1’

Chapter 8

REMAINING TASKS.

8.1 Overview

Our project FISPAN has by far approximately been 40% completed. We aim to complete the remaining within a month from now.

8.2 To-Do

1. Though we have manually collected 1000 images per alphabet, we are yet to process those images and use them to train our model.
2. Also, we are yet to collect data for dynamic gestures.
3. Since we have used only 30 data per gesture, we have used LSTM as our model. BUt after we will process all our data, we shall be using VGG-19 model for training.
4. We aim to add text to speech feature in our project.
5. The accuracy is by far very low from the threshold. We aim to extend the accuracy above 95%.

Chapter 9

BIBLIOGRAPHY

[1] A. Thakur, P. Budhathoki, S. Upreti, S. Shrestha, and S. Shakya, “Real Time Sign Language Recognition and Speech Generation,” *Journal of Innovative Image Processing*, vol. 2, no. 2, pp. 65–76, Jun. 2020, DOI: 10.36548/jiip.2020.2.001.

[2] J. Zheng *et al.*, “An Improved Sign Language Translation Model with Explainable Adaptations for Processing Long Sign Sentences,” *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1–11, Oct. 2020, DOI: 10.1155/2020/8816125.

[3] M. M. Rahman, M. S. Islam, M. H. Rahman, R. Sassi, M. W. Rivolta and M. Aktaruzzaman, "A New Benchmark on American Sign Language Recognition using Convolutional Neural Network," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-6, DOI: 10.1109/STI47673.2019.9067974.

[4] V. I. Pavlovic, R. Sharma, and T. S. Huang, “Visual interpretation of hand gestures for human-computer interaction: a review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, Jul. 1997, DOI: 10.1109/34.598226.

[5] Jiyong Ma, W. Gao, Jiangqin Wu, and Chunli Wang, “A continuous Chinese sign language recognition system,” *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, 2000, DOI: 10.1109/afgr.2000.840670.

[6] C. Vogler and D. Metaxas, “Handshapes and Movements: Multiple-Channel American Sign Language Recognition,” *Gesture-Based Communication in Human-Computer Interaction*, pp. 247–258, 2004, DOI: 10.1007/978-3-540-24598-8_23.

[7] E. Abraham, A. Nayak, and A. Iqbal, “Real-Time Translation of Indian Sign Language using LSTM,” *2019 Global Conference for Advancement in Technology (GCAT)*, Oct. 2019, DOI: 10.1109/gcat47503.2019.8978343.

[8] World Health Organization, WHO. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-andhearing-Loss

[9] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way,” *Towards Data Science*, Dec. 15, 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[10] O. Boufeloussen, “Simple Explanation of Recurrent Neural Network (RNN),” *The Startup*, Sep. 05, 2020. https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363

[11] C. Olah, “Understanding LSTM Networks -- colah’s blog,” *Github.io*, Aug. 27, 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[12] Jason Brownlee, “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning,” *Machine Learning Mastery*, Jul. 02, 2017. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/