

TRIBHUVAN UNIVERSITY

Department of Computer science and Information Technology

Madan Bhandari Memorial College

Sign Language Recognition

By:

Ayush Jung Karki (15963/074)

Binita Subedi (15965/074)

Shreedhar Pandeya (15993/074)

A PROJECT PROPOSAL TO THE DEPARTMENT OF COMPUTER SCIENCE AND

INFORMATION TECHNOLOGY IN PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR THE BACHELOR’S DEGREE IN BSC. CSIT

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

KIRTIPUR KATHMANDU, NEPAL

SEPTEMBER, 2021

# DECLARATION

Project entitled “**Sign Language Recognition System**” which is being submitted to Madan Bhandari Memorial College, Department of Computer Science and Information Technology, Binayaknagar, Kathmandu, Nepal for the fulfillment of the seventh semester under the supervision of **Mr Phul Babu Jha.**

This project is original and has not been submitted earlier in part or full in this or any other form to any university or institute, here or elsewhere, for the award of any degree.  
  
  
By:

Ayush Jung Karki (15963/074)

Binita Subedi (15965/074)

Shreedhar Pandeya (15993/074)

# RECOMMENDATION

This is to recommend that **Shreedhar Pandeya, Binita Subedi, Ayush Jung Karki** have carried out research entitled **“Sign Language Recognition System”** for the fulfillment of the seventh semester in **Bachelor of Science in Computer Science and Information Technology** under my supervision. To my knowledge, this work has not been submitted for any other degree.

They have fulfilled all the requirements laid down by the Madan Bhandari Memorial College, Department of Computer Science and Information Technology, Binayaknagar, Kathmandu.

…………………………

Mr Phul Babu Jha

Project Supervisor

Department of Computer Science and Information Technology, Madan Bhandari Memorial College

Binayaknagar, Kathmandu, Nepal

**Madan Bhandari Memorial College**

**Affiliated to Tribhuwan University**

**CERTIFICATE OF APPROVAL**

The undersigned certify that they have read and recommended to the Department of Computer Science and Information Technology, IOST, Madan Bhandari Memorial Campus, a project report entitled “Sign Language Recognition System” submitted by Shreedhar Pandeya (15993), Binita Subedi (15965), and Ayush Jung Karki (15963). The project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled, and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of a Bachelor’s degree in Computer Science and Information Technology.

|  |  |
| --- | --- |
| **Signature of the Supervisor** | **Signature of the HOD/Coordinator** |
| **Signature of the External Examiner** | **Signature of the Internal Examiner** |

# ABSTRACT

Every normal human being sees, listens, and reacts to surrounding. There are some unlucky individuals who does not have this important blessing. Such individuals, mainly deaf and dumb, they depend on communication via sign language to interact with others. However, communication with ordinary individuals is a major impairment for them since not every typical people comprehend their sign language. Furthermore, this will cause a problem for the deaf and dumb communities to interact with others, particularly when they attempting to involve into educational, social and work environments. In this project, the objectives are to develop a sign language translation system in order to assist the hearing or speech impaired people to communicate with normal people.

# TABLE OF CONTENTS

TITLE PAGE i

[ABSTRACT ii](#_Toc7242)

[Table Of Contents iii](#_Toc7243)

[1 INTRODUCTION 1](#_Toc7244)

[1.1 Motivation 1](#_Toc7245)

[1.2 Objectives 2](#_Toc7246)

[1.3 Problem statement 2](#_Toc7247)

[2 METHODOLOGY 3](#_Toc7248)

[2.1 Software Development Approach](#_Toc7249) 3

[2.2 Requirement Analysis](#_Toc7250) 3

[2.2.1 Literature Review 3](#_Toc7251)

[2.3 Requirement Specification 4](#_Toc7253)

[2.3.1 Functional Requirements 4](#_Toc7254)

[2.3.2 Non Functional Requirements 5](#_Toc7255)

[2.4 Feasibility Study 6](#_Toc7256)

[2.4.1 Technical 6](#_Toc7257)

[2.4.2 Economic 6](#_Toc7258)

[2.4.3 Operational 6](#_Toc7259)

[2.5 Working Mechanism of Purposed System 7](#_Toc7260)

[2.6 Scheduling 8](#_Toc7261)

[2.6.1 Gantt Chart 8](#_Toc7262)

[2.7 System Flow Chart 9](#_Toc7263)

[3 TOOLS AND TECHONOLOGIES 10](#_Toc7264)

[3.1 Python 10](#_Toc7265)

[3.2 Numpy 10](#_Toc7266)

[3.3 TensorFlow and Keras 10](#_Toc7267)

[3.4 pyttsx3 10](#_Toc7268)

[4 Expected Outcome 11](#_Toc7269)

[REFERENCES 12](#_Toc7270)

# INTRODUCTION

Each individual utilize language to communicate with others. The listening to disabled individuals likewise utilize language to communicate among themselves. Sign language is basically utilized by hearing impaired people to communicate with each other, developed by deaf communities. Communication through signing is a very organized nonverbal language using both non-manual and manual correspondence. Non manual signals are essentially outward appearance, head movement, stance and orientation of the body. While manual signals incorporate movement and orientation of hand that passes on typical significance .

On the other hand, communication with typical individuals is a major impediment for them since not every ordinary people comprehend their gesture based communication. To beat this issue, sign language recognition system is expected to assist the deaf and mute people to communicate with normal people.

## Motivation

Each ordinary individual sees, tunes in, and responds to encompassing. Notwithstanding, there are some less blessed individuals who are denied of this important blessing. Such individuals, mainly deaf and dumb, they depend on sign language to communicate with others. Statistic shows that about 9 billion people in this world are deaf and dumb. Interactions between deaf-dumb people and normal people have always been a troublesome assignment.

This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exists in higher rates among the deaf population, especially when they are immersed in a hearing world [1]. Generally, not every ordinary people comprehend the communication through sign language utilized by the weakened. This makes an almost no space for them as communication is one of the necessities of life. Furthermore, this will cause a problem for the deaf and dumb communities to interact with others, particularly when they attempting to coordinate into instructive, social and workplaces. To conquer this issue, a sign language recognition system must be developed with a specific end goal to kill the imperative between the ordinary and debilitated individual.

## Objectives

The main goal of this project is to develop sign language translation system that can translate the sign language into text. Since not every typical people being educate with communication through signing, this system will help them to comprehend the language of deaf and dumb people so that will give points of interest to them in conducting their daily tasks ahead. This project also aims to meet the following objectives:

* To develop gesture recognizing system that can recognize sign gesture of Sign Language and translate it into text.
* To test the accuracy of the system.

## Problem statement

Despite the fact that the deaf and dumb people can impart without issues amongst themselves, there is serious challenge for the hearing impaired person trying to communicate with normal people. This is because not every single typical people can comprehend their gesture based communication. The greater part of ordinary individuals has not been taught about the sign language. As communication is imperative, this issues inevitably makes a limitation for the impaired individuals to correspond with the normal. Therefore, a sign language translator must be developed to tackle those issues.

# METHODOLOGY

## Software Development Approach

We’ll start with the most basic aspects and work our way up from there, adding new features as we go. For the development process, we will use an iterative approach, mostly Agile technique as there might come up new ideas that may arise along the development pathway.

## Requirement Analysis

### Literature Review

**Deaf Mute Communication Interpreter- A Review [2]** : This paper aims to cover the various prevailing methods of deaf-mute communication interpreter system. The two broad classification of the communication methodologies used by the deaf –mute people are - Wearable Communication Device and Online Learning System. Under Wearable communication method, there are Glove based system, Keypad method and Handicom Touch-screen. All the above mentioned three sub-divided methods make use of various sensors, accelerometer, a suitable micro-controller, a text to speech conversion module, a keypad and a touch-screen.

**An Efficient Framework for Indian Sign Language Recognition Using Wavelet Transform [3]:**The proposed ISLR system is considered as a pattern recognition technique that has two important modules: feature extraction and classification. The joint use of Discrete Wavelet Transform (DWT) based feature extraction and nearest neighbor classifier is used to recognize the sign language.

## Requirement Specification

### Functional Requirements

Functional requirements define the basic system behavior. Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs.

Although the fact that two people are involved, only one of them interacts directly with the system. This is the hearing impaired person that performs sings in front of the system responds by translating them to speech. The other person can be considered as passive user since it does interacts directly with the system, but only listens/reads to recognized signs.

Requirements can be represented by describing the interaction that different actors (users, external systems) have with the system and where the:

1.System (presented as a black box)

2. Hearing Impaired(person that performs the signs)

3. Normal hearing(passive user of the system)

Therefore the system requirements can be specified as :

* Capture the real time Sign Language performed by impaired person.
* Predict the provided Sign language and convert to Text.
* Dictating the Text to Speech.

### Non Functional Requirements

While functional requirements define what the system does or must not do, non-functional requirements specify how the system should do it. Non-functional requirements do not affect the basic functionality of the system . Even if the non-functional requirements are not met, the system will still perform its basic purpose. The Non Functional Requirements for Sign Language Recognition System are :

1. Clean UI

The user interface will be clean and beginners friendly.

1. Security

The application will not store any data extracted form document permanently.

1. Performance

Application will be fast and fetching of the data will take less time

1. Maintenance

Timely updates will be provided for bug fixes, and adding new features.

1. Reliability

The features, information and data provided will be reliable and authentic.

## Feasibility Study

A feasibility study helps to determine the benefit of proposed system to the targeted users. It also determines if the proposed system can be built successfully with the cost, resources available, time and effort. The study is conducted by studying the collected requirements.

### Technical

All the tools, software and resources needed to develop Sign Language Recognition System are easily available in internet. Our core functionality will be written in Python . It can be done easily with proper guidance of Supervisor.

### Economic

Sign Language Recognition System will be cheap to build. The application requires just a Python developer to develop .There are not so much difficulty in regards to cost management.

### Operational

All the features of the application are possible to create with out much hassle.

## Working Mechanism of Purposed System

The system will be developed to be primarily run on desktop environment , we will try to integrate the software to mobile platform if possible.

The general operational flow of system will be as:

1. Creating the Image Dataset and Preprocessing the data

We will obtain the most of Dataset on the internet , and we will create our own dataset too. For creating our own dataset, We will be having a live feed from the video cam and every frame that detects a hand in the ROI (region of interest) created will be saved in a directory

1. Training a CNN on captured Data:

We design a CNN using python and train the above dataset. We

load the data using ImageDataGenerator of Keras and train the Dataset. After training the CNN we obtain a model that will extract the certain features from the dataset and helps to classify the gestures.

1. Predict the Data and Classify the Gesture.

Using above model we try similar actions on real time image/video

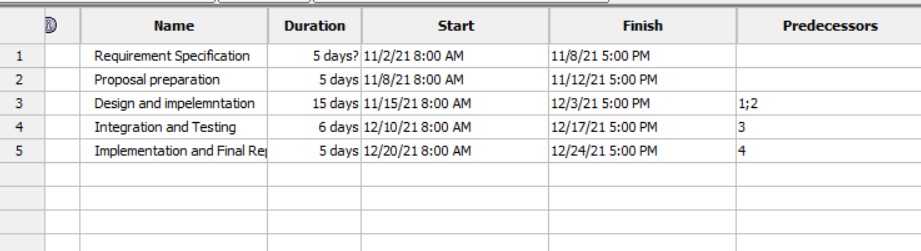
data to predict the data and classify the gesture and present Text Output accordingly.

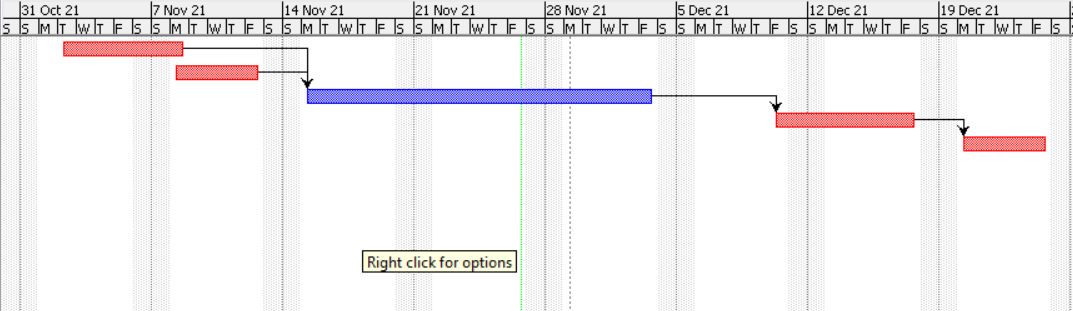
1. Text to Speech

We will be using pyttsx3 [3] for converting above obtained Text to Speech.

## Scheduling

### Gantt Chart





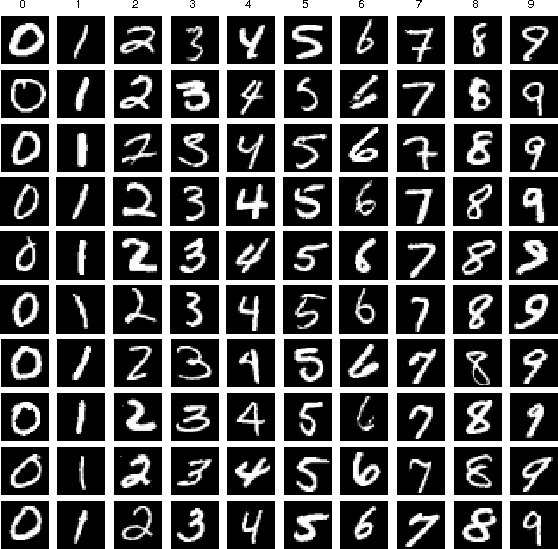
## Flow Chart

## arabic sign language dataset

# System Design

## Phases in Optical Character Recognition

### Data collection



**Figure 10: Datasets**

First of all, we got our datasets from two different sources:

* + - 1. Standard MNIST 0-9 dataset
      2. Kaggle A-Z dataset

The standard MNIST 0-9 dataset can be seen on the left side of Figure 10. The MNIST dataset contains 0-9 handwritten digits, each of which is contained in a 28 x 28 grayscale image. It has a training set of 60,000 examples, and a test set of 10,000 examples.

The Kaggle A-Z dataset can be seen on the right side of Figure 8. The Kaggle A-Z dataset contains of A-Z capital letters taken from NIST Special Database 19 and rescales them to be 28 x 28 grayscale image, so it is in the same format as our MNIST data. It consists of 370,000+ handwritten alphabets in CSV format.

### Loading the datasets

To train the model, first of all we will have to load the datasets into our program that we have previously collected. To load the MNIST 0-9 dataset, we can simply use the keras library to import the mnist module and then call its load\_data method. Likewise, to load the Kaggle A-Z dataset, first of all we have to download the dataset file which is in CSV

format and place it in the project folder. Then, we process and receive the data and labels from it.

### 5.2 Initializing the model

The parameters we used for training the model are as follow:

* + - 1. Epochs: 50
      2. Initial Learning Rate (LR): 0.1
      3. Batch Size (BS): 128

We partition the data into training and testing splits using 80% of the data for training and the remaining 20% for testing purposes. We construct our CNN architecture using the SGD optimizer and a standard learning rate decay schedule. The first three parameters of CNN’s build technique show that each character/digit is represented as a 32x32 pixel grayscale picture. Then, we compile our model with the "categorical crossentropy" loss and the SGD optimizer we've used before.

## Training the model

We train our model using the model.fit method. The following are the parameters:

* aug.flow: It establishes inline data augmentation
* validation\_data: Tests input images (testX) and test labels (testY).
* steps\_per\_epoch: The number of batches that are run for each pass of the entire training data.
* epochs: The number of complete runs over the entire data set during training.
* class\_weight: Weights due to the imbalance of data samples for various classes.
* verbose: Displays a training progress bar.

Following that, we assigned labels to each individual character. Then, we concatenated all of our digits and letters to generate an array with a single digit or number as each element. We generated predictions on the test set and produced our classification report to evaluate our model.

### Saving the model

Now, we need to save the model that includes the architecture and final weights after we've completed our training. Our model will be saved to disk as a Hierarchical Data Format version 5 (HDF5) file.

### Plotting the graph

Next, we used matplotlib’s plt to generate a line plot for the training loss and validation set loss along with titles, labels for the axes, and a legend. The data for the training and validation losses were taken from H's history, the results of the fit method from the model, with one point for each epoch. The training loss curves plot was stored as plot.png.

### Preprocessing the user input

We loaded the input image and converted it to grayscale, and then used Gaussian blurring to reduce the noise. Following that, we used the Canny module of cv2 library to detect the borders of our blurred image. We used contour detection to find the contours for each character. Then, we sorted the contours from "left to right".

### Detecting characters

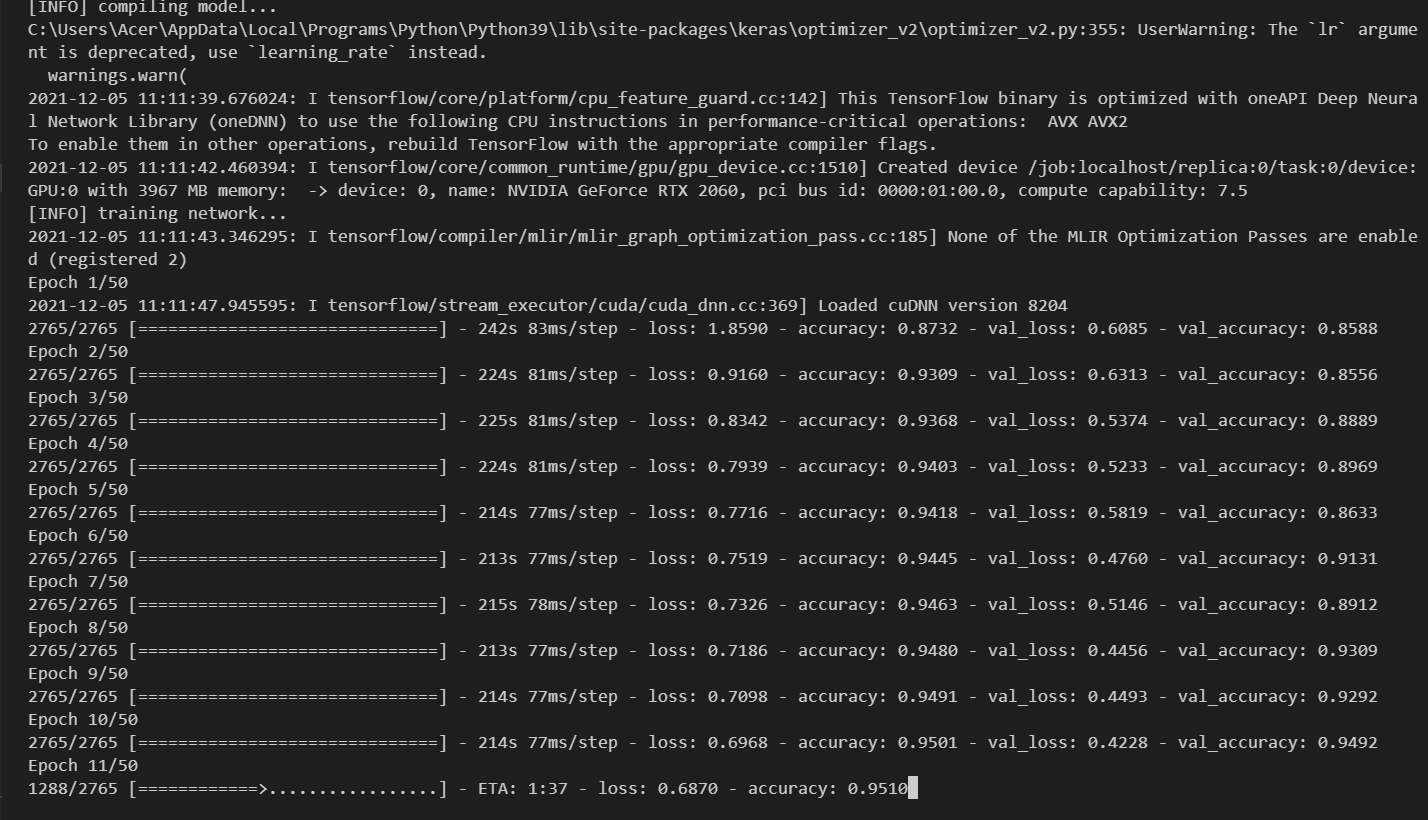
We looped over each contour and performed a series of four steps:

* + - 1. Step 1: Select and extract appropriate-sized contours.
      2. Step 2: Using a thresholding algorithm, clean up the images with the goal of having white characters on a black background.
      3. Step 3: Resize each character to a 32x32 pixel picture with a border.
      4. Step 4: Prepare each padded ROI for character classification.

### Predicting and displaying the result

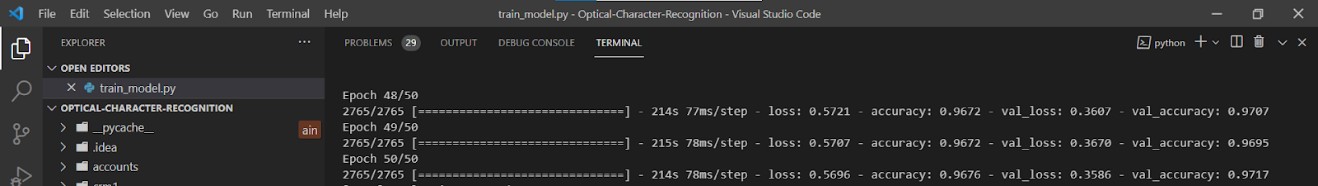
Firstly, we extracted the original bounding boxes with associated chars in NumPy array format. Then, we classified the entire batch of pre-processed characters with the model.predict method, which resulted in a list of predictions. We looped over each prediction and corresponding bounding box. We grabbed the highest probability prediction resulting in the character's label inside the loop. We annotated each character with the bounding box and label text in order to visualize the results, and then displayed the result.

# CHAPTER 5 IMPLEMENTATION AND TESTING



### Figure 11: Training Model Epoch 1-11

As seen from the above figure 11, our model had less accuracy and more loss. In the epoch 1, the training loss was 1.8590 and the training accuracy was 87.32%, meanwhile the testing loss was 0.6085 and the testing accuracy was 85.88%. Then, gradually training our model until epoch 10, the statistics were improved i.e. training loss was 0.6968 and training accuracy was 95.01%, meanwhile the testing loss was 0.4228 and testing accuracy was 94.92%. Clearly, the statistics obtained up to epoch 10 was not enough and the model certainly had more room for improvements.



### Figure 12: Training Model Epoch 48-50

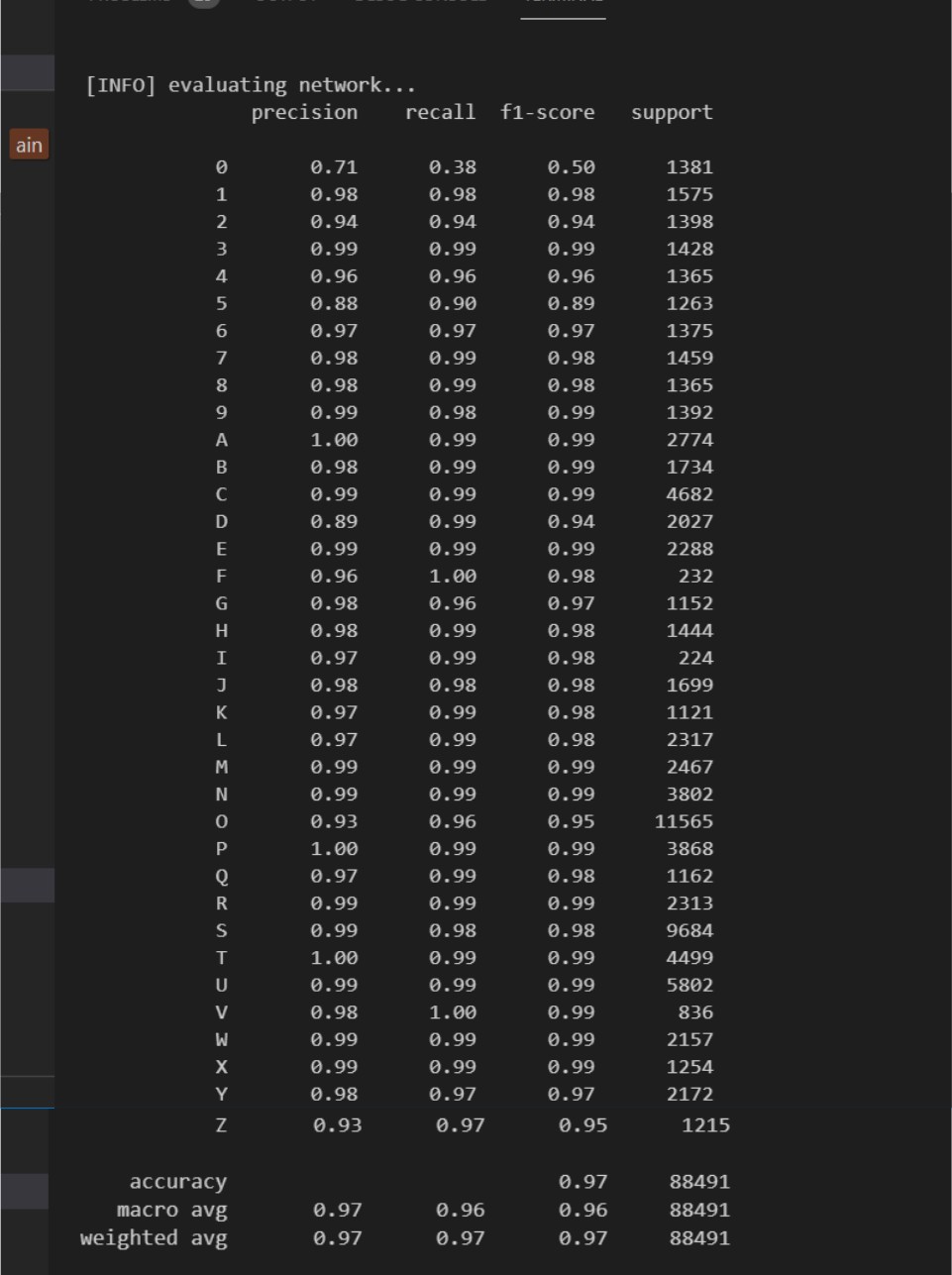
After training our model up to epoch 50 as seen from figure 12 our statistics were further improved. The training loss was 0.5696 and training accuracy was 96.76%, meanwhile the testing loss was 0.3586 and the testing accuracy was 97.17%. Clearly, there wasn’t much room for improvement by increasing the epoch size after 50 epochs as the statistics were

not improving much in the latter cycles. Also, there was a chance of overfitting if we had trained our model for more epoch cycles, which would decrease the accuracy of our model.



### Figure 13: Loss/Accuracy vs. Number of Epoch

From the plot in figure 13, we can clearly visualize the changing statistics of our model along with increase in number of epoch. It shows little sign of overfitting, which means that our model is performing well. We can also see that the training and testing accuracy is increasing and loss is decreasing from the graph.



### Figure 14: Classification report of all characters

In figure 14, the performance measures of all the characters that have been tested using our model can be seen. All the scores obtained of the various performance measures (precision, recall, f1-score, support) for all the characters look pretty good. It can concluded that the accuracy of 97% is obtained in our model.



### Figure 15: Sample output from testing set

The above figure 15 represents the sample output from the testing set of data. The OCR model is performing well, but not perfect. Our model seems to be confused between handwritten characters that appear to be similar to other characters. For example: In the third row fourth column, the actual character was ‘0’ but our model predicted it as ‘O’, because both of the characters look very similar, even to the human eyes.

## 

# TOOLS AND TECHONOLOGIES

## Python

The main functionality of the project will be coded python

### TensorFlow[4] and Keras[5]

We will use TensorFlow for all AI related tasks and Deep Learning .

### Numpy[6]

Numpy will be used for mathematical and logical operations.

### Pyttsx3[7]

We will use Pyttsx3 specifically for converting text into speech

3.1.4 OpenCV

A python library that handles real-time computer vision

# Expected Outcome

The expected product we are planning to obtain is a Desktop application that will have an basic user interface where input data(Sign language Gestures) will be captured and an interface that will display Text output and provide Audio Output.

# References

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2093933/

[2] Sunitha K. A, Anitha Saraswathi.P, Aarthi.M, Jayapriya. K, Lingam Sunny, “Deaf Mute Communication Interpreter- A Review”, International Journal of Applied Engineering Research ,Volume 11, pp 290-296 ,2016.

[3] Mandeep Kaur Ahuja, Amardeep Singh, “Hand Gesture Recognition Using PCA”, International Journal of Computer Science Engineering and Technology (IJCSET ), Volume 5, Issue 7, pp. 267-27, July 2015.

[4] <https://www.tensorflow.org/api_docs>

[5] <https://keras.io/api/>

[6] <https://numpy.org/doc/>

[7] https://pypi.org/project/pyttsx3/