Architecture

Computer Vision Solution for Hearing-Impaired

Written By	Niranjan Shrestha, Prajin Bajracharya
Document Version	0.1
Last Revised Date	24 March 2022

Document Version Control

Change Record:

Version	Date	Author	Comments
0.1	24 – March-2022	Niranjan Shrestha, Prajin Bajracharya	Introduction and Architecture Defined

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments
0.1				

Table of Contents:

Abstract

Sign language is a way of communication by people suffering from hearing loss. Around 360 million people globally suffer from disabling hearing loss out of which 328 million are adults and 32 million children. Thus, with increasing number of people with deafness, there is also a rise in demand for translators. Minimizing the communication gap between hearing impaired and normal people becomes a necessity to ensure effective communication among all. To develop a communication approach for healthy people to understand sign language used by hearing impaired people with ease, in this paper we propose a method of converting the cue symbols (ASL Gestures) to speech. We are in the verge of developing an application named “Computer Vision Solution for Hearing Impaired” which is a project intended for helping every person to learn and understand sign language to make conversation easier and understandable for people. Although there are lots of application that can convert text to speech but only few places to learn sign language. Sign language conversion is a technique used for converting the hand gestures into their respective voice or text message. Different software can be used for this type of conversion like MATLAB, LABVIEW, C programming etc. But we used Python 3.0 as our core language for programming. We are using tensor flow as a machine learning library to process the sign language images as well as their meaning and for the detection of the sign language as input and automated speech as output.. To sum up, this application is helpful for each and every person willing to have conversation with the disabled person, this project is solely for mute person to converse with normal person using the modern technology to convert their sign language to automated speech.

Keywords: Sign language, tensor flow, disability, Python 3.0

1. Introduction

1.1 Why this Architecture Design Document?

The purpose of this document is to provide a detailed architecture design of the Computer Vision Solution for Hearing Impaired. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli and reactions. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher management for its approval.

The main objective of this project is to provide the platform to interpret the series of signs (gestures) to speech so as to make conversation easier and meaningful for people with disability.

1.2 Scope

This system will be deployed in Web application. This system will be employed in the modern devices that can read the sign language to interpret it into automated speech. The conversation experience will be improved between people (with mute and normal person). All people will be familiarized with sign language. The system will be applicable in various government and non-government organizations.

1.3 Constraints

We only predict the 26 letters and some of the words but not the whole sentence. However, by combination of those words and letters we can able to predict different simple sentences easily.

2. Technical Specifications

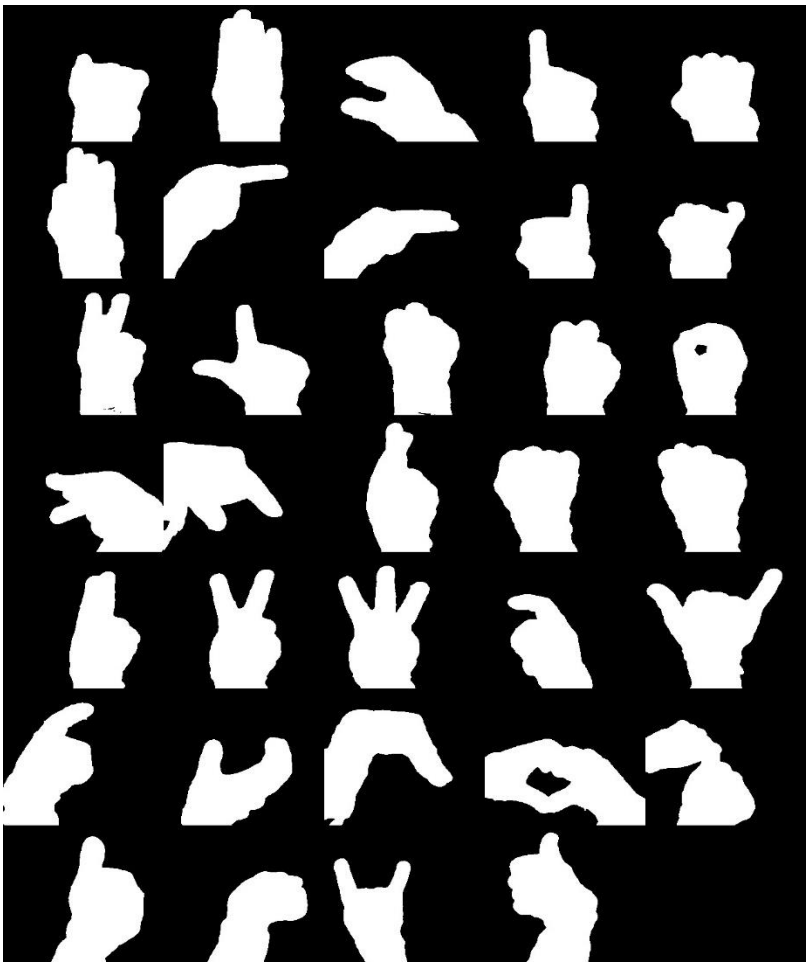
2.1 Dataset Overview

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. Data sets can also consist of a collection of documents or files.

For the dataset, we managed to capture the images of following terminologies and made the dataset for this project:

1. 26-class fingerspelling recognition
2. 2-extra function recognition
3. 6-word class fingerspelling recognition

For the project we would like to use to data sets of sign languages. Some images of datasets of sign languages are shown below:



2.2 Logging

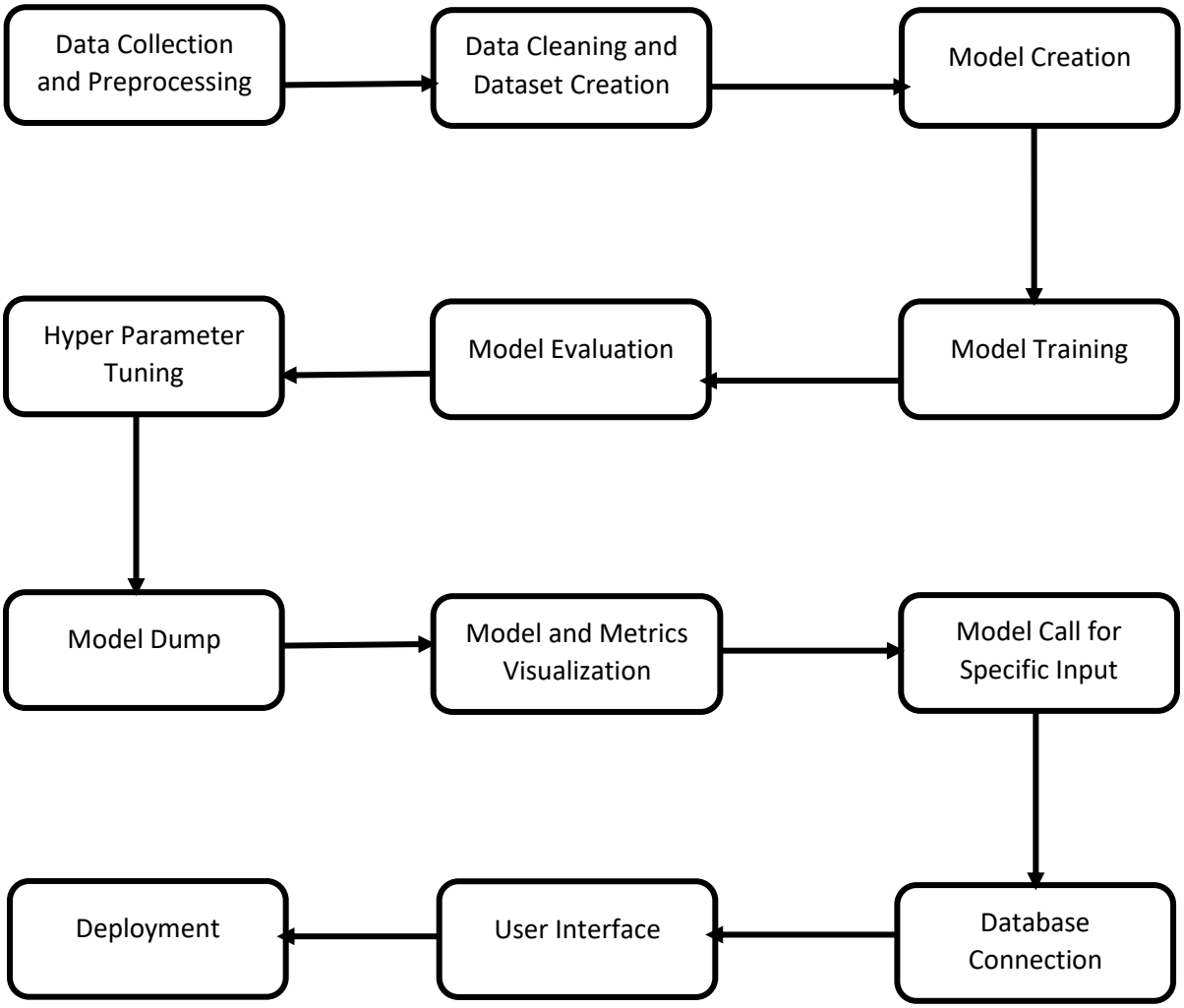
In this Project we are logging every process so that user will know what process is running internally.

Step-By-Step Description:

- In this Project we defined logging for every function, class.

- By logging we can monitor every insertion, every flow of data in database.
- By logging we can monitor every step which may create problem or every step which is important in file system.
- We have designed logging in such a way that system would not hang even after so many logging's so that we can easily debug issues which may arise during process flow.
- With use of logging we can check the status of model is loaded or not

3. Architecture



4. Architecture Description

4.1 Data Collection

Since the dataset provided by kaggle, google and any other sites aren't that appropriate for our model. So, we decided to create a simple python program that can capture desired number of images from a real time video through the use of webcam for each letters or information and separate each of them in folder. However for that, we have used various functions of OpenCV which are listed below:

- Transformation
- Smoothing
- Edge Detection
- Segmentation
- Thresholding
- Contours Detection

4.2 Data Cleaning/ Data Transformation

In the Cleaning process, the captured images are represented in in the form of array as well as appropriate label are also converted in the form of array and those array of images and labels didn't needed any kind of additional change or transformation because while capturing those images it had captured appropriate images following the methods and operations of OpenCV. So for the data validation we created the pickle file which contained dataset that was splitted in different dataset with different numbers such as training, testing and validation dataset so that it could be further processed in the neural net.

4.3 Model Creation

Creating the convolutional neural network model involved making choices about various parameters and hyper-parameters. We used TensorFlow as main python library to implement CNN We took decisions about the number of layers to use in our model. The model used 8*2 layers including number of repeated max pooling and convolution layer.

In first step we have used the layer Conv2d which is a 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of

outputs. Thus, we have used 16 filter along with kernel size of 2*2 with input shape of 256*256*1 dimension where none determines the parameter can be in variable state with activation function as relu. Then after this we have used Maxpooling2D with pool size of 2*2 pooling windows along with 2*2 strides with padding of same size. Similarly the Conv2d and Maxpooling 2d is repeated again 2 times with 32 and 64 filters along with same activation function and also pooling size 3*3,5*5 and strides also 3*3 , 5*5 with the padding same respectively. Then the output from here goes to the flatten layer which reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension. Thus, the output goes to the dense layer containing 128 neurons which feeds back the output to the neurons with activation function relu. Then the output from it goes to the dropout layer which select 20 % of the neurons and set their weights to zero. At last, we use again the dense layer with number of classes we have used that is 34 as neurons to thus help reach the weights to appropriate class.

4.4 Model Training

After we have created our model, we simply created an instance of the model and fit it with our training data. The biggest consideration when training a model is the amount of time the model takes to train. To consume less time, we used various parameters for training the model. Such as we used Adam as an optimizer, learning rate as 3e-4, loss as categorical cross entropy etc. We can specify the length of training for a network by specifying the number of epochs to train over. Thus, we specified 15 epochs. The longer we train a model, the greater its performance will improve, but too many training epochs and you risk overfitting. But we were able to train for limited time only due to lack of time and for risk of overfitting we used dropout layer as well. Currently we used only 26 letters, 2 special function and 6 words for training each with dataset about 2000.

4.5 Model Evaluation

For model evaluation, the model's performance was compared against a validation dataset, a data set that the model hasn't been trained on. The model's performance was compared against this validation set and analyzed its performance through different metrics. The most common metric is "accuracy", the amount of correctly classified images divided by the total number of images in your data set. After viewing the accuracy of the model's

performance on a validation dataset, we trained the network again using slightly tweaked parameters. Finally, the network's performance was tested on a testing set.

4.6 Hyper parameter Tuning

For Hyper Parameter tuning, we used Manual Search algorithm because it was easy for us changing the parameter manually and by changing little we got he required point where bias was low and variance was low according to bias-variance tradeoff.

4.7 Model Dump

After comparing all accuracies and checking the ROC as well as AUC curve accuracy we saved or dumped the model in the form of .h5 extension which is regarded as default model extension from keras.

4.8 Model and Metrics Visualization

Model Visualization provides reason and logic behind to enable the accountability and transparency on the model. Machine Learning models considered as Black Box Models due to complex inner workings. Data Scientists deliver a model with high accuracy. There are some scenarios where models cannot be explained to the public because the system may hack. For the Model Visualization, we used confusion matrix as it is regarded as one of the main method for model visualization.

Visualization metrics can be defined as the metrics that are calculated to measure the attributes and capture the properties of visualization, to extract the meaningful information of data. We used python library known as Tensorboard to analyze and visualize the different metrics such as loss and accuracy of Training and Validation phases.

4.9 Model call for Specific input

Model Call is done by loading the model and opening the web cam. Then hand signs were waved in the dedicated Region of Inference (ROI) so that how much the model is accurate. After the model evaluation, we used OpenCV to capture the real time video through various OpenCV programming codes to create a window that can capture the real time video along with ROI (Region of Interest) where the hand sign is predicted from that area only and we also used the methods of OpenCV just like in data collection which are:

- Transformation

- Smoothing
- Edge Detection
- Segmentation
- Thresholding
- Contours Detection

Thus these methods help to increase more accuracy of predicting the sign of the hand. We also created another window to show the letter that has been detected. Currently due to low accuracy from training we were able to predict these hand signs:

- 26-class fingerspelling recognition
- 2-extra function recognition
- 6-word class fingerspelling recognition

4.10 Database Connection

For the different purposes of User Interface such as login, signup and other authorization processes we used Cassandra database for storing the emails, usernames and passwords of various user so that our web app is much more secure and much more interactive. So for the initiation we made a database named flask in datastax astra which creates databases that runs Multi-cloud DBaaS Built on Apache Cassandra. After that, we wanted to connect the flask database that we named, to our flask application. Hence, we connected the database using the steps mentioned in datastax astra website with proper guidance and steps in it. Then after successful connection we used CQLAlchemy one of the library of Flask for helping Cassandra database synchronization in flask environment. Then we used varieties of functions related to CQLAlchemy for getting and sending the data into and from the database respectively. The database mainly contains following type of data:

SN	Records	Type	Key
1	id	UUID	Primary
2	first_name	Text	-
3	last_name	Text	-
4	username	Text	Primary
5	email	Text	Primary
6	password	Text	Primary

4.11 User Interface

Flask is a web framework. This means flask provides us with tools, libraries and technologies that allow us to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website.

So to access our system through a web page we have integrated our system to the flask frontend as flask is the web framework of python as well so to implement the system build in python wasn't quite difficult. So our flask web app is made simple which consists of simple home page, login page, signup page, profile page and camera page hosted at port 8000. We have loaded the model 'handsign.h5' with the help keras library successfully in flask frontend. Then we have created all the templates, interfaces, small database using CQLAlchemy for the web app. Also we have created various functions for login, signup, etc. Finally to get the result from the camera page, we have converted the OpenCV camera output to an image bytes and hence it becomes suitable to be implemented in the flask frontend and thus real time video can be generated in the camera page. Thus the camera will detect the Hand signs and display result in same camera page.

4.12 Deployment

As it was difficult for deploying our model in the docker or any other cloud deployment apps as it required camera to load through OpenCV and required heavy resources with paid version that we couldn't afford it, so we deployed in the local host for our ease. Due to these undeniable circumstances we were compelled to deploy in local host as output with a proper justification of our system design.