# AARC – A Sign Language Interpreter

**A Project Report**

*Submitted by*

**Adit Nair**

**Aditya Nair**

**Rishabh Nanawati**

**Christopher Paralkar**

*Under the Guidance of*

**Prof. Ameyaa Biwalkar**

*in partial fulfillment for the award of the degree of*

**Bachelors of Technology**

**Computer Engineering**

At


**Mukesh Patel School of Technology, Management & Engineering**

**Narsee Monjee Institute of Management Studies, Mumbai**

**MARCH 2020**

# Contents

# CERTIFICATE

This is to certify that the project entitled AARC – A Sign Language Interpreter is the bonafide work carried out by Adit Nair, Aditya Nair, Rishabh Nanawati and Christopher Paralkar of B. Tech, MPSTME (NMIMS), Mumbai, during the VI semester of the academic year 2019-2020, in partial fulfillment of the requirements for the Course Programming Language.

_____

Prof. Ameyaa Biwalkar

Internal Mentor

_____ _____

Examiner 1 Examiner 2

# DECLARATION

We, Adit Nair, Aditya Nair, Rishabh Nanawati and Christopher Paralkar, Roll No. E002, E003, E004, E013 B. Tech. (Computer Engineering), Semester VI understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source: IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Names: Adit Nair, Aditya Nair, Rishabh Nanawati, Christopher Paralkar

Roll Nos.: E002, E003, E004, E013

Place: Mumbai

Date: 2nd April, 2020

# ACKNOWLEDGEMENT

We would like to thank our faculty Prof. Ameyaa Biwalkar, Prof. Shubha Puthran and Dr. Dhirendra Mishra for their contributions in the form of teaching, feedbacks and suggestions in the fields of Python, Software Engineering and Image Processing respectively.

# Introduction

Communication between people often occurs using the means of speech, text or signs/gestures. In person, two or more abled people are able to communicate through the means of speech while two or more hearing impaired or deaf individuals are able to communicate through the means of sign language or gestures. However, an abled person and a hearing impaired or deaf individual do have not any common means of communication except text. In person this form of communication seems tedious and cumbersome. As a result, a means of communication needs to be developed in order to bridge the gap of communication between an abled person and a differently abled person. This project focuses on a sign language interpreter interpreting American Sign Language using the help of webcam and machine learning. The output would be text by which the abled person can understand what the differently abled person is trying to say through sign language.

A Sign language interpreter is someone who helps hearing impaired or deaf individuals understands a spoken language by converting it into sign language during one on one conversation. However sign language interpreters cannot be available 24/7 so a technological means should be available to overcome this conundrum and provide a convenient way of interpreting sign language. The prototype implemented in this project uses American Sign Language as its data set for interpretation. It uses image processing techniques for capturing and detecting the hand gestures or sign language while using machine learning algorithms in order to train the model and using the trained model to interpret the American Sign Language.

Our problem consists of four tasks to be done such as:

1. Obtaining the video input of the user signing
2. Classifying each sign made to a letter by training the model.
3. Displaying the letter closely related to the one signed by the user.
4. Creating a custom gesture.

Image processing techniques like contouring were used for detecting the hand.

## Literature review

In the number of papers that we have reviewed, it is noted that the use of ASL was predominant. When we researched upon this it is because ASL does not have many hand-to-body gestures as the other sign languages do, hence making it easy for the interpreter to operate on and the algorithm to recognise.

Brandon Garcia and Sigberto Alarcon Viesca of Stanford University attained a validation accuracy of nearly 98% with five letters and 74% with ten letters using Convolutional Neural Networks along with American Sign Language as a Data set.

We used convolutional neural networks to achieve recognition of signs. This stems from the fact that CNNs learn features as well as the weights corresponding to each feature. Furthermore, our aim was to use more of image processing rather than data processing unlike other methods like Bayesian networks which use Hidden Markov Models that need models to be clearly defined a priority.

To reduce time taken to train and test data, we also implemented a simpler model to predict the label from the input features. This model calculates the cosine difference between the input features and the stored features. The label of the feature with the least cosine difference is given out as the prediction.

# Software and APIs Used

The entire project was implemented via Anaconda Windows Environment on a system with 8GB RAM and Intel® Core™ i5-8250U CPU @1.60GHz. The system used python3.6 along with various libraries (licensed either GNU, MIT or BSD) that have been discussed further.

## OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

In this project, OpenCV was used to capture images, apply image processing techniques and identify features of the processed image.

## NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
In this project, NumPy is used to store image matrices and feature data in the form of 2D matrices.

## Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

In this project, Keras was used in developing Convolution Neural Networks (CNNs) of multiple layers to store feature data of dataset images and predict the label of the input image.

## Tensorflow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

In this project, Tensorflow was used to convert 1D image data into 2D data which can be used in CNNs to predict the label of the input image

## Tkinter

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk.
In this project, Tkinter was used to provide a GUI interface to the whole project.

# Methods Implemented

The algorithm can be broken down into two parts – Image Processing and Model Training. The algorithm as described in the flowchart works as follows: -

- Raw Image is captured from a video stream. This video stream is provided by the camera input of the system.
- The raw image is cropped to focus only on the hand. Thus, a region of interest (ROI) is obtained.
- Background Subtraction is performed to remove unwanted data from the ROI. This reduces the computational complexities and prevents any other object in the ROI from being considered as a key feature. The ROI now has 90% less unwanted data.
- Skin Masking is applied to identify the hand as the only object and any other object, which may have been considered as foreground in the previous step, is identified and removed. The ROI now has 95% less unwanted data.
- Edge Detection is applied to generate the boundary line of the hand and highlight the particular gesture that is being made. The pixel values of these edge-detected images are stored as 1D numpy arrays.
- The 1D arrays in the dataset made needs to be converted into 2D arrays to be able to work with convolution neural networks (CNN). For example, an image of 784 pixel values becomes an image of size 28x28.
- Data structures called Tensors are created to store the features (x) and their corresponding labels (y). For example, the features of an image of hand gesture for the letter "A" would be stored in x while the character "A" will be stored as y.
- The following hyperparameters are set:
  - epoch: It basically tells how much the network is going to train.
  - batch_size: It is the number of training samples that will be fed to the network in one iteration.
  - learning_rate: It defines how fast will the network adjust and, well, learn.

- The architecture being used for this algorithm is covnet. Convnet is part of deep, feed-forward artificial neural networks that can perform a variety of task with even better time and accuracy than other classifiers.
- The network is then trained and evaluated. The result is then analysed.

While a lot of the code has been built from scratch, a few library functions have been used throughout the code that have aided in the working of the project.

## OpenCV

This module provided various methods to implement the image processing techniques that went into enhancing the hand gesture present in the input image.

`createBackgroundSubtractor()`: This is a method provided to eliminate background and retain only the foreground of the image. It was used to remove unwanted parts of the region of interest (ROI) and focus only on the hand gesture.

`MedianBlur()`: This is method that applies medium blur transformation to the given input image. This method was used in the difference of blur technique to highlight the edges.

`Canny()`: This is a method that applies Canny Edge Detection technique on the image to highlight the edges around separate objects based on their contrast ratios. It was used to identify and retain only the edges of the hand gesture.

## KAZE

This is a module provided by the OpenCV library that is used in feature extraction. It housed various functions like detect(), drawKeyPoints(), compute(), flatten() that helped in detection and visualization of key points and computation and flattening of the data held by these key points.

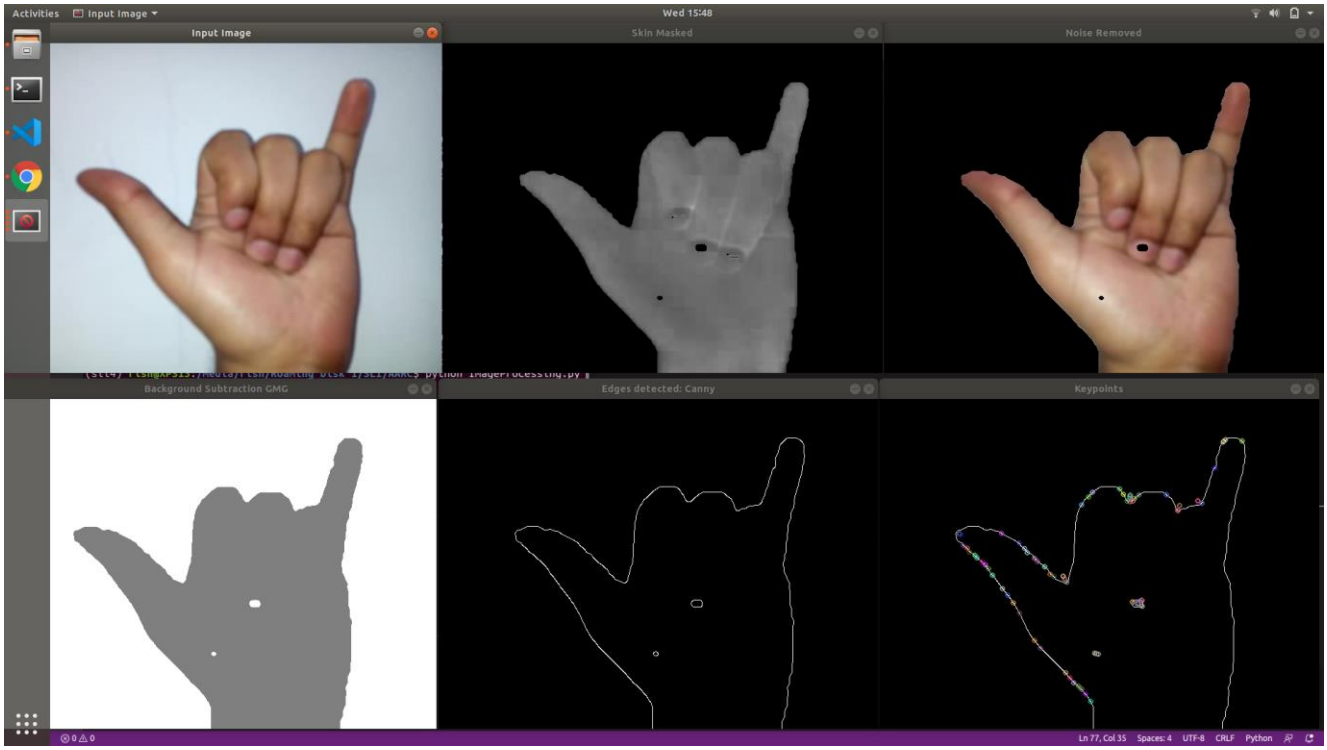
## Keras & Tensorflow

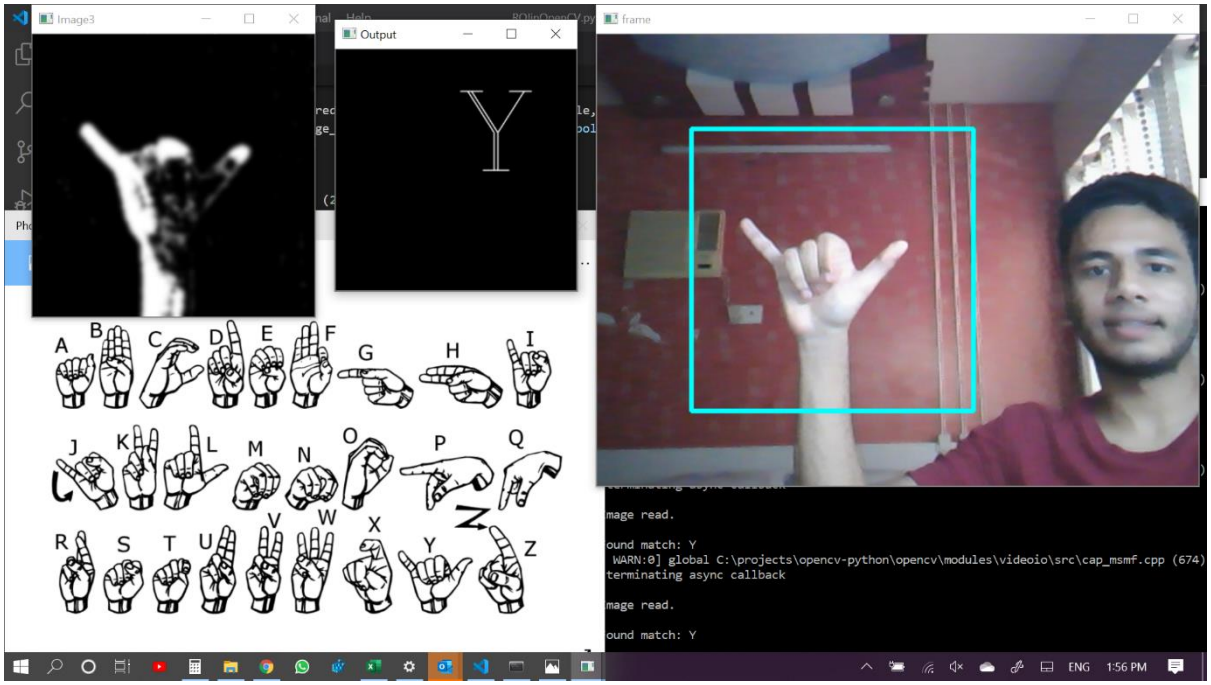
Tensorflow helped in converting the 1D array of features into 2D arrays called “Tensors”. These tensors were passed through the CNN CovNet architecture that consisted of 6 layers, comprising multiple layers of Conv2D, MaxPool2D, a Dropout Layer and 2 linear layers to convert the 2D arrays back into 1D arrays.

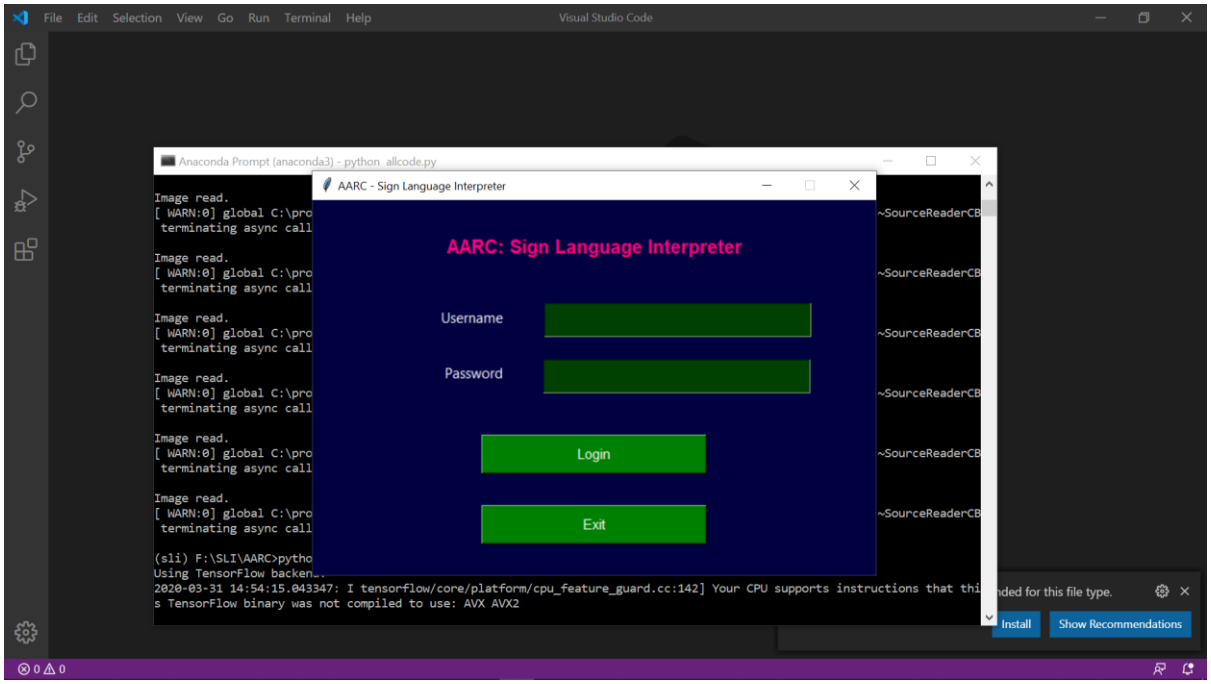
## Tkinter

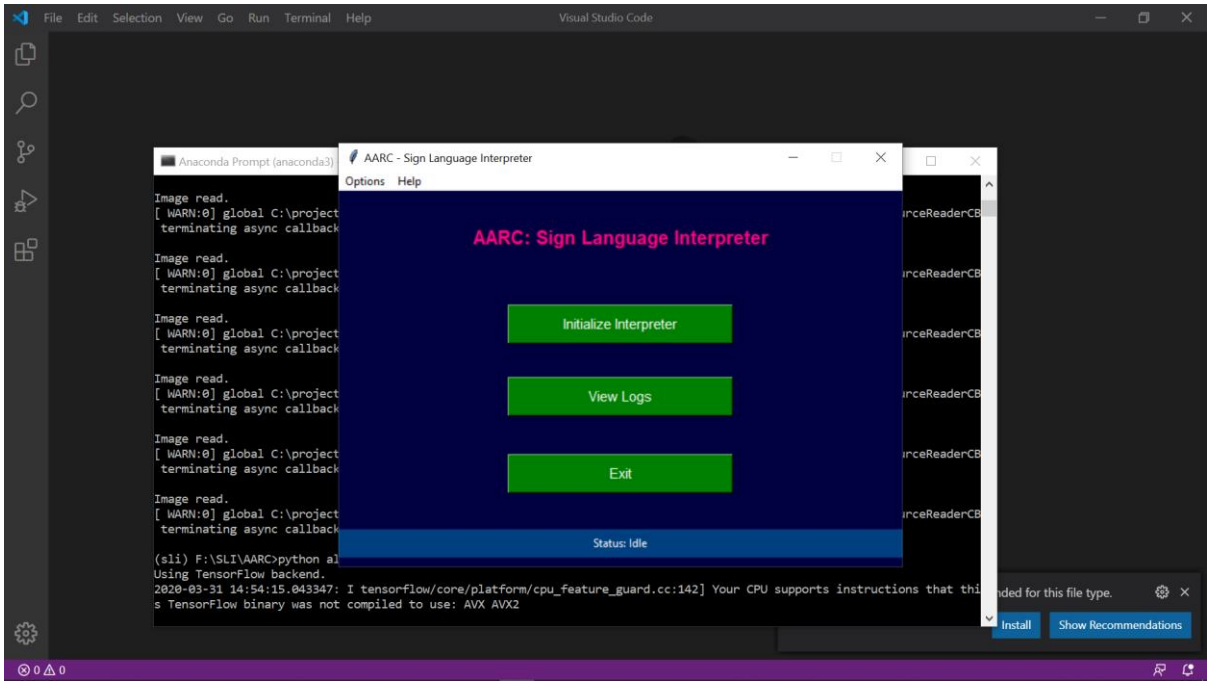
Many methods of tkinter, like StringVar(), mainloop(), etc. were used to implement GUI designs that made user interaction smooth.

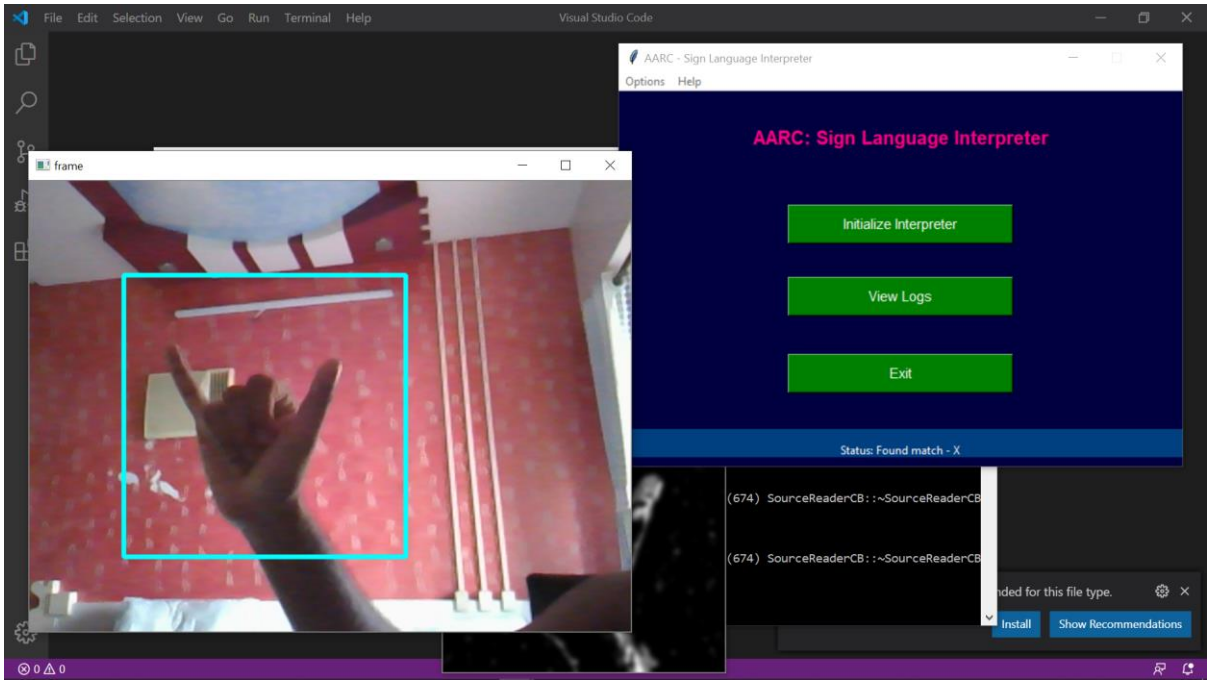
# Screenshots

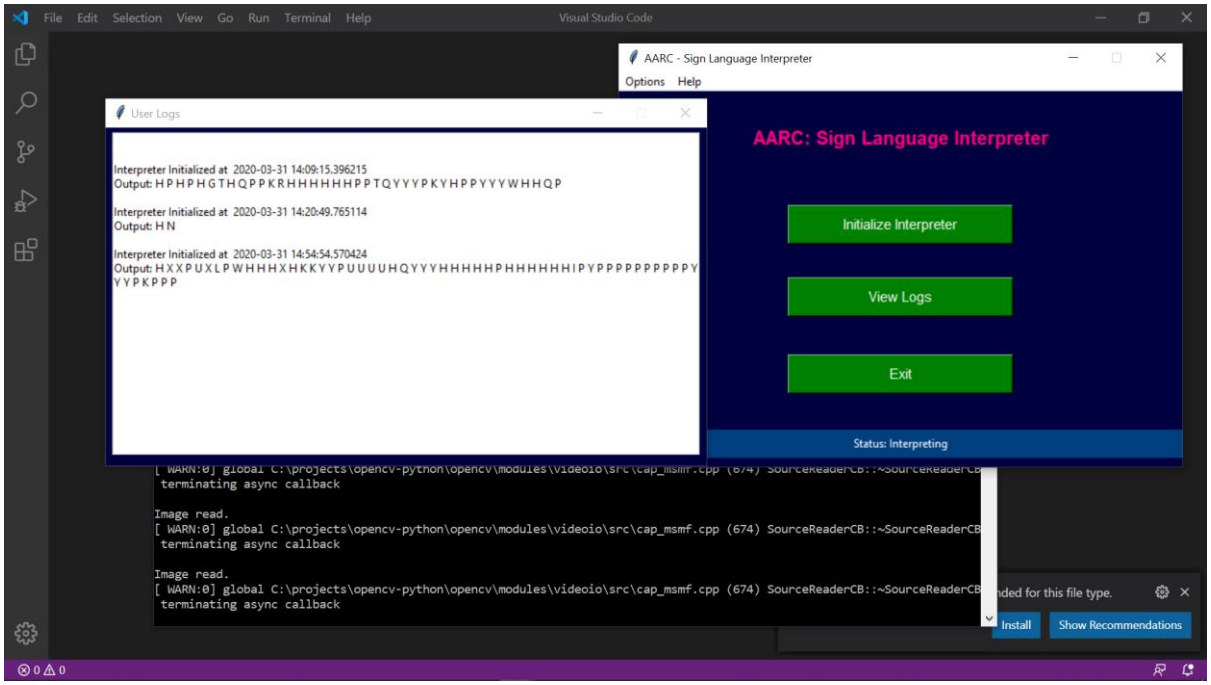


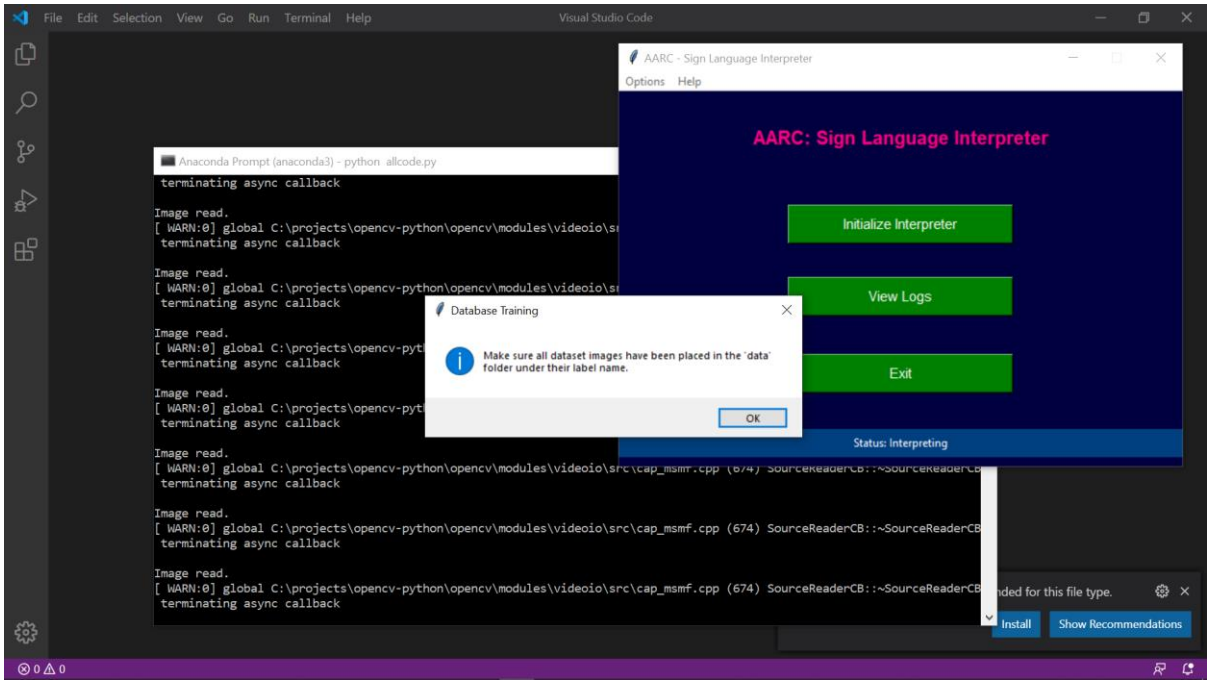



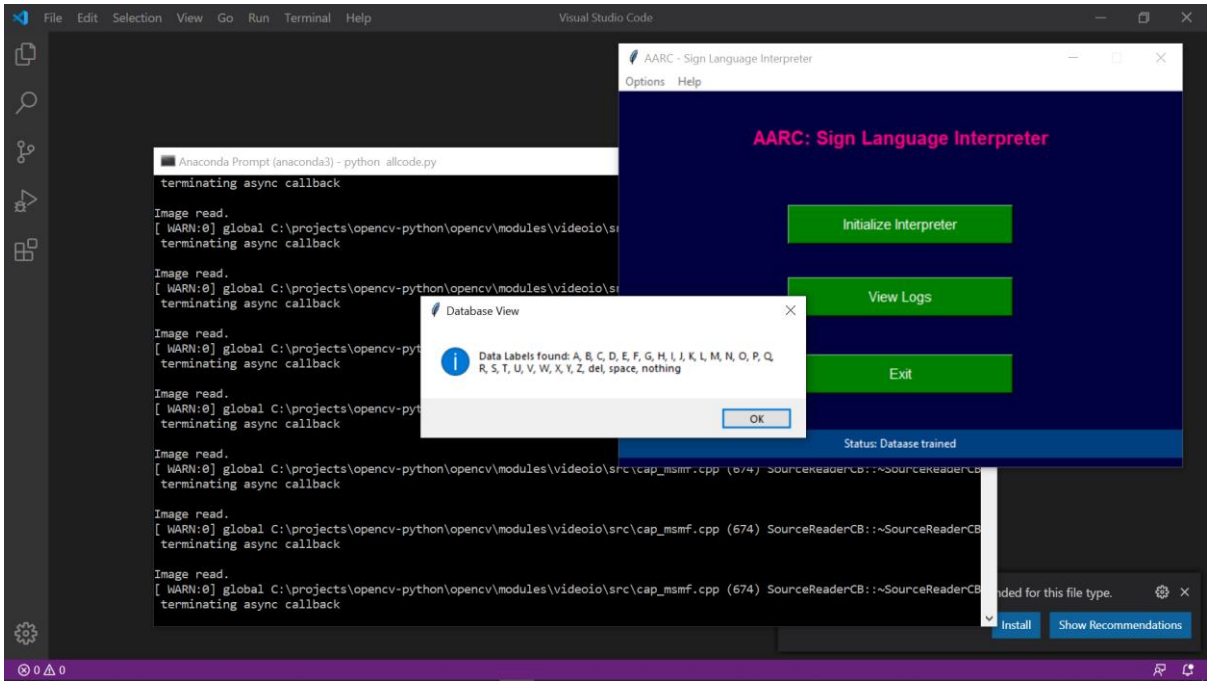

File Edit Selection View Go Run Terminal Help
Visual Studio Code
Anaconda Prompt (anaconda3) - python allcode.py
AARC - Sign Language Interpreter
AARC: Sign Language Interpreter
Username
Password
Login
Exit
Image read.
[ WARN:0] global C:\pro
terminating async call
(sli) F:\SLI\AARC>pytho
Using TensorFlow backen
2020-03-31 14:54:15.043347: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX AVX2
Install
Show Recommendations

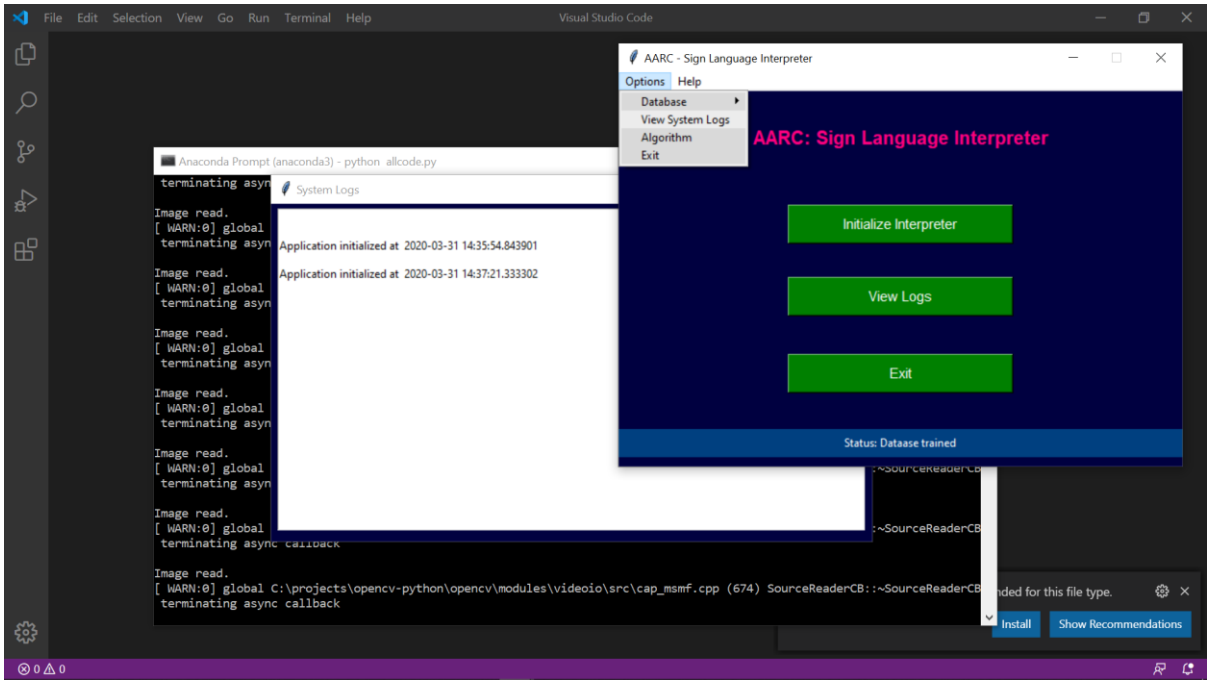

File Edit Selection View Go Run Terminal Help
Visual Studio Code
Anaconda Prompt (anaconda3)
AARC - Sign Language Interpreter
Options Help
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Idle
Image read.
[ WARN:0] global C:\project
terminating async callback
(sli) F:\SLI\AARC>python al
Using TensorFlow backend.
2020-03-31 14:54:15.043347: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX AVX2
Install
Show Recommendations

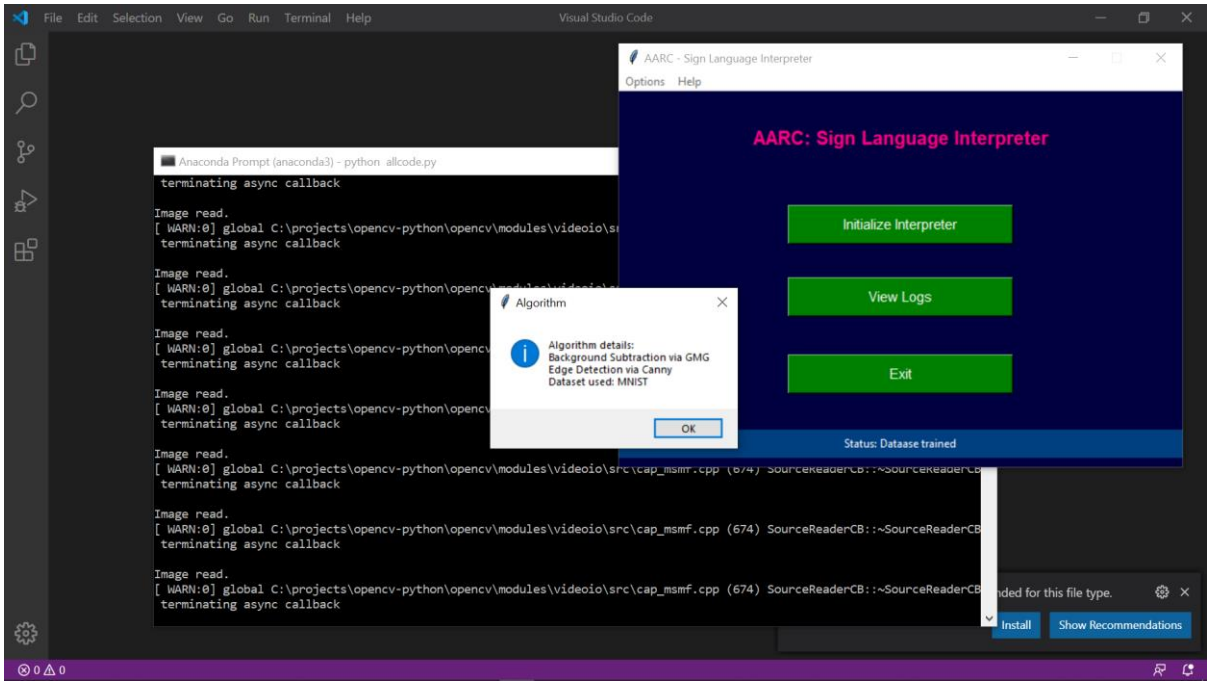

File Edit Selection View Go Run Terminal Help
Visual Studio Code
frame
AARC - Sign Language Interpreter
Options Help
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Found match - X
(674) SourceReaderCB::~SourceReaderCB
(674) SourceReaderCB::~SourceReaderCB
Install
Show Recommendations

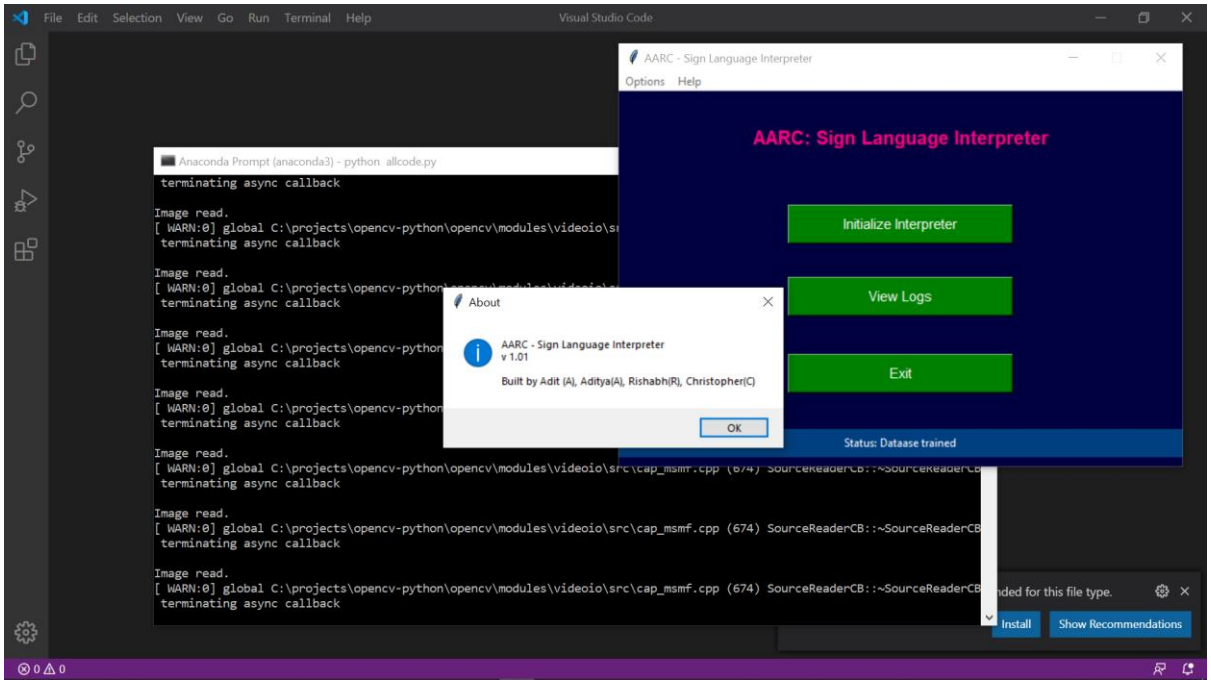

File Edit Selection View Go Run Terminal Help
Visual Studio Code
AARC - Sign Language Interpreter
Options Help
User Logs
Interpreter Initialized at 2020-03-31 14:09:15.396215
Output: H P H P H G T H Q P P K R H H H H H H P P T Q Y Y Y P K Y H P P Y Y Y W H H Q P
Interpreter Initialized at 2020-03-31 14:20:49.765114
Output: H N
Interpreter Initialized at 2020-03-31 14:54:54.570424
Output: H X X P U X L P W H H H X H K K Y Y P U U U U H Q Y Y Y H H H H H P H H H H H H I P Y P P P P P P P P P P Y
Y Y P K P P P
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Interpreting
terminating async callback
Image read.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
terminating async callback
Image read.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
terminating async callback
Install
Show Recommendations


File Edit Selection View Go Run Terminal Help
Visual Studio Code
AARC - Sign Language Interpreter
Options Help
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Interpreting
Anaconda Prompt (anaconda3) - python allcode.py
terminating async callback
Image read.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
Database Training
Make sure all dataset images have been placed in the `data` folder under their label name.
OK
Install
Show Recommendations


File Edit Selection View Go Run Terminal Help
Visual Studio Code
AARC - Sign Language Interpreter
Options Help
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Dataase trained
Anaconda Prompt (anaconda3) - python allcode.py
terminating async callback
Image read.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
Database View
Data Labels found: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, del, space, nothing
OK
Install
Show Recommendations


File Edit Selection View Go Run Terminal Help
Visual Studio Code
AARC - Sign Language Interpreter
Options Help
Database
View System Logs
Algorithm
Exit
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Dataase trained
Anaconda Prompt (anaconda3) - python allcode.py
System Logs
Application initialized at 2020-03-31 14:35:54.843901
Application initialized at 2020-03-31 14:37:21.333302
terminating async callback
Image read.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
Install
Show Recommendations


File Edit Selection View Go Run Terminal Help
Visual Studio Code
AARC - Sign Language Interpreter
Options Help
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Dataase trained
Anaconda Prompt (anaconda3) - python allcode.py
Algorithm
Algorithm details:
Background Subtraction via GMG
Edge Detection via Canny
Dataset used: MNIST
OK
terminating async callback
Image read.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
Install
Show Recommendations


File Edit Selection View Go Run Terminal Help
Visual Studio Code
AARC - Sign Language Interpreter
Options Help
AARC: Sign Language Interpreter
Initialize Interpreter
View Logs
Exit
Status: Dataase trained
About
AARC - Sign Language Interpreter
v 1.01
Built by Adit (A), Aditya(A), Rishabh(R), Christopher(C)
OK
Anaconda Prompt (anaconda3) - python allcode.py
terminating async callback
Image read.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
Install
Show Recommendations

# Conclusion and Future Scope

This project can be tied in as a module with a host of applications. It may be loaded into an IoT device such as an Arduino or Raspberry Pi and be used a portable interpreter that provides smooth communication between the abled and the verbally impaired. It can also be used for video calling applications, where the video feed is input and the predicted letters are typed using a ducky script. New gestures might be included in the dataset that not only symbolise other characters such as numbers, punctuations and emojis but also be used as a touch free device where a particular gesture can act as an event and trigger a particular action/activity.

# Societal Applications

It goes without saying that the benefits of an online sign language interpreting tool are clearly aimed towards the differently abled. It was one of our main aims to do something with a huge societal impact and this is why we selected this topic to work on. In many parts of the world, a sort of unintentional ostracization of the deaf and dumb occur owing to only a small number of people being able to understand them. This digital interpreter can be applied to reduce the gap between the differently and fully abled, enabling the former to communicate freely with the latter, thus opening themselves up to be understood and appreciated just as any regular human being would.