

SIGN LANGUAGE TO TEXT AND SPEECH TRANSLATION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

JEEVANA SHREE M (17SCS5077)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of ENIYA M (37110211) and JEEVANA SHREE M (37110289) who carried out the project entitled "**SIGN LANGUAGE TO TEXT AND SPEECH TRANSLATION USING MACHINE LEARNING**" under my supervision from November 2020 to April 2021.

Internal Guide

Mrs. DEEPA D, M.E., (Ph.D.)

Head of the Department

Dr. Vigneshwari. S, M.E., Ph.D.

Dr. Lakshmanan. L, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **ENIYA M**, hereby declare that the Project Report entitled **SIGN LANGUAGE TO TEXT AND SPEECH TRANSLATION USING MACHINE LEARNING** done by me under the guidance of **Mrs. DEEPA.D, M.E., (Ph.D.)** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering / Technology degree in Computer Science and Engineering.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneshwari M.E., Ph.D.** and **Dr. L. Lakshmanan M.E., Ph.D., Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. DEEPA.D, M.E., (Ph.D.)** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Sign language is one of the most seasoned and most regular type of language for correspondence, however since the vast majority don't know communication via gestures and mediators are hard to obtain, we have thought of a continuous strategy utilizing neural organizations for fingerspelling based American gesture-based communication. In our technique, the hand is first gone through a channel and after the channel is applied the hand is gone through a classifier which predicts the class of the hand signals. Our technique gives exactness to the 26 letters of the letter set. We use CNN calculation which is not difficult to utilize. A Convolutional Neural Network is a Deep Learning calculation which can take in an information picture, allocate significance (learnable loads and predispositions) to different viewpoints/objects in the picture and have the option to separate one from the other. Motions are the nonverbally traded messages and these signals are perceived with vision. This nonverbal correspondence of hard of hearing and unable to speak individuals is called gesture- based communication.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 CAUSES	2
	1.2 MOTIVATIONS	2
	1.3 USAGE	3
	1.4 CLASSIFICATIONS	4
2	LITERATURE SURVEY	5
3	METHODOLOGY	9
	3.1 SOFTWARE AND HARDWARE REQUIREMENTS	9
	3.1.1 SOFTWARE REQUIREMENTS	9
	3.1.1.1 PYCHARM	9
	3.1.1.2 KERAS	9
	3.1.1.3 TENSORFLOW	10
	3.1.1.4 OPENCV	11
	3.1.2 HARDWARE REQUIREMENTS	12
	3.2 OVERVIEW OF THE PLATFORM	12
	3.2.1 PYTHON	12
	3.2.2 WHY PYTHON?	13
	3.3 NEURAL NETWORKS	14
	3.3.1 ADVANTAGES	14
	3.3.2 APPLICATIONS	15
	3.4 CONVOLUTIONAL NEURAL NETWORKS	16

	3.4.1 LAYERS OF CNN	17
	3.5 PROPOSED SYSTEM	18
	3.6 SYSTEM ARCHITECTURE	19
	3.6.1 MODULE DESCRIPTION	19
	3.7 DATASET GENERATION	20
	3.8 GESTURE CLASSIFICATION	21
	3.8.1 LAYER 1	21
	3.8.2 LAYER 2	22
	3.9 IMPLEMENTATION	23
	3.9.1 AUTOCORRECT FEATURES	23
	3.9.2 TRAINING AND TESTING	24
4	RESULTS AND DISCUSSIONS	25
	4.1 RESULTS	25
	4.2 DISCUSSIONS	26
5	CONCLUSION AND FUTURE SCOPES	28
	5.1 CONCLUSIONS	28
	5.2 SUMMARY	29
	5.3 FUTURE SCOPE	29
	REFERENCES	31
	APPENDIX	32
	A. SOURCE CODE	32
	B. SCREENSHOTS	38
	C. PLAGIARISM REPORT	41

LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
3.1	NEURAL NETWORK ARCHITECTURE	16
3.2	CNN ARCHITECTURE	18
3.3	SYSTEM ARCHITECTURE	20
3.4	DATASET	21

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
ANN	Artificial Neural Network
NN	Neural Network
ASL	American Sign Language
SLR	Sign Language Recognition

CHAPTER 1

INTRODUCTION

American sign language is a predominant sign language. Since the only disability D&M people have is communication related and they cannot use spoken languages, hence the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behaviour and visuals. Deaf and dumb (D&M) people make use of their hands to express different gestures to express their ideas with other people. In our project we basically focus on producing a model which can recognise Fingerspelling based hand gestures in order to form a complete word by combining each gesture. The gestures we aim to train are as given in the image below.

Wherever communities of deaf people exist, sign languages have developed as useful means of communication, and they form the core of local Deaf cultures. Although signing is used primarily by the deaf and hard of hearing, it is also used by hearing individuals, such as those unable to physically speak, those who have trouble with spoken language due to a disability or condition or those with deaf family members, such as children of deaf adults. It is unclear how many sign languages currently exist worldwide. Each country generally has its own native sign language, and some have more than one. The 2020 edition of Ethnologue lists 144 sign languages, while the SIGN-HUB Atlas of Sign Language Structures lists over 200 of them and notes that there are more which have not been documented or discovered yet. Some sign languages have obtained some form of legal recognition. Linguists distinguish natural sign languages from other systems that are precursors to them or obtained from them, such as invented manual codes for spoken languages, home sign, "baby sign", and signs learned by non-human primates.

Different sign languages are speculating in particular areas. For a case, British Sign Language (BSL) is an entirely different language from an ASL, and USA people who familiarise with ASL would not easily understand BSL. Some nations adopt capabilities of ASL of their sign languages. Sign language is a way of verbal

exchange via human beings diminished by speech and listening to loss. Around 360 million human beings globally be afflicted via unable to hearing loss out of which

328000000 are adults and 32000000 children. Hearing impairment extra than 40 decibels in the better listening to ear is referred as disabling listening to loss. Thus, with growing range of people with deafness, there is moreover a rise in demand for translators. Minimizing the verbal exchange gap among listening to impaired and regular humans turns into a want to make certain effective conversation among all.

1.1 CAUSES:

Wherever communities of deaf people exist, sign languages have developed as useful means of communication, and they form the core of local Deaf cultures. Although signing is used primarily by the deaf and hard of hearing, it is also used by hearing individuals, such as those unable to physically speak, those who have trouble with spoken language due to a disability or condition (augmentative and alternative communication), or those with deaf family members, such as children of deaf adults.

It is unclear how many sign languages currently exist worldwide. Each country generally has its own native sign language, and some have more than one. The 2020 edition of *Ethnologue* lists 144 sign languages, while the SIGN-HUB Atlas of Sign Language Structures lists over 200 of them and notes that there are more which have not been documented or discovered yet.

Some sign languages have obtained some form of legal recognition. Linguists distinguish natural sign languages from other systems that are precursors to them or obtained from them, such as invented manual codes for spoken languages, home sign, "baby sign", and signs learned by non-human primates.

1.2 MOTIVATIONS:

For interaction between normal people and D&M people a language barrier is created as sign language structure which is different from normal text. So, they depend on vision-based communication for interaction. If there is a common interface that

converts the sign language to text the gestures can be easily

understood by the other people. So, research has been made for a vision-based interface system where D&M people can enjoy communication without really knowing each other's language. The aim is to develop a user-friendly human computer interfaces (HCI) where the computer understands the human sign language. There are various sign languages all over the world, namely American Sign Language (ASL), French Sign Language, British Sign Language (BSL), Indian Sign language, Japanese Sign Language and work has been done on other languages all around the world.

Using a qualitative approach known as the Critical Incident Technique (CIT), faculty and staff were asked to reflect on their sign language learning experiences, and their responses were examined for motivational patterns. Principal motivating factors were intrinsic in nature, including a desire to perform well in one's position, personal goals, and an interest in sign language per se. Integrative factors were also important, especially an interest in social interactions with deaf people.

1.3 USAGE:

In such communities' deaf people are generally well integrated in the general community and not socially disadvantaged, so much so that it is difficult to speak of a separate "Deaf" community. Many Australian Aboriginal sign languages arose in a context of extensive speech taboos, such as during mourning and initiation rites. They are or were especially highly developed among the Warlpiri, Warumungu, Dieri, Kaytetye, Arrernte, and Warlmanpa, and are based on their respective spoken languages.

A pidgin sign language arose among tribes of American Indians in the Great Plains region of North America (see Plains Indian Sign Language). It was used by hearing people to communicate among tribes with different spoken languages, as well as by deaf people. There are especially users today among the Crow, Cheyenne, and Arapaho. Unlike Australian Aboriginal sign languages, it shares the spatial grammar of deaf sign languages. In the 1500s, a Spanish expeditionary,

Cabeza de Vaca, observed natives in the western part of modern- day Florida using sign language, and in the mid-16th century Coronado mentioned that communication with the Tonkawa using signs was possible without a

translator. Whether or not these gesture systems reached the stage at which they could properly be called languages is still up for debate. There are estimates indicating that as many as 2% of Native Americans are seriously or completely deaf, a rate more than twice the national average. Sign language is also used by some people as a form of alternative or augmentative communication by people who can hear but cannot use their voices to speak.

1.4 CLASSIFICATIONS:

Although sign languages have emerged naturally in deaf communities alongside or among spoken languages, they are unrelated to spoken languages and have different grammatical structures at their core. Sign languages may be classified by how they arise.

In non-signing communities, home sign is not a full language, but closer to a pidgin. Home sign is amorphous and generally idiosyncratic to a particular family, where a deaf child does not have contact with other deaf children and is not educated in sign. Such systems are not generally passed on from one generation to the next. Where they are passed on, creolization would be expected to occur, resulting in a full language. However, home sign may also be closer to full language in communities where the hearing population has a gestural mode of language; examples include various Australian Aboriginal sign languages and gestural systems across West Africa, such as Mofu-Gudur in Cameroon.

A village sign language is a local indigenous language that typically arises over several generations in a relatively insular community with a high incidence of deafness, and is used both by the deaf and by a significant portion of the hearing community, who have deaf family and friends. The most famous of these is probably the extinct Martha's Vineyard Sign Language of the US, but there are also numerous village languages scattered throughout Africa, Asia, and America.

CHAPTER 2

LITERATURE SURVEY

Malchanov et al. [1] have using machine learning algorithms presented the idea of a translation with skin colour tone to detect the ASL. They have made a skin colour segmentation that automatically depicts the colour and give the tune to it for further detection. They have used YCbCr spacing of colour as it's vastly used in video template code and gives the efficient results of colour tone for human skin. Further they have taken the CbCr plane to distribute the skin tone colour. People from Different ethnicity have their tones different which is crafted in a model. Deep Learning methods and several machine learning algorithms are used to demonstrate translator to translate between parties.

Pigou et al. [2] explained, the lack of automated structure to translate symptoms from LSM makes integration of listening to-impaired human beings to society extra difficult. This painting affords a totally new technique for LSM alphanumerical signs popularity based on 3D Haar-like features extracted from depth pictures captured by means of the Microsoft Kinect sensor. Features are processed with a boosting set of policies. To observe normal performance of our technique, we Identified a tough and speedy sign from letters and numbers, and in comparison, the results with the use of traditional 2D Haar-like capabilities. Our gadget is capable of recognize static LSM signs and signs with a higher accuracy percentage than the one obtained with extensively used 2D features. As It has been preferred for a society that having a sign language for hearing impaired and deaf people to communicate.

Liu et al. [3] depict the usage of new and standardized model of communication system in which it basically targets the deaf people as that they have further explained. The system comprised of the following two scenarios like at first, Spanish speeches to Spanish sign translation that makes the usage of a translator to break up the words and have them converted into the stream set of signs which in along with each other makes a sentence and also depicts in an avatar form. Now the second scenario, the sign language that from a speech was generated then send to the generator, what this generator does that it makes out those signs into the Spanish

word which will make sense which basically converts the sign language

into the Spanish based spoken language words. Now whatever the data previously was generated is comprises of words, now the last step is to convert the words into a sentence that is word to speech conversion. This paper consists of real example in city of Spain, Toledo that involved government personal and the deaf people. Further the paper describes the possible outcome and the scope of it in the future.

Husang et al. [4] it depicts the communication between deaf people and the other parts of society. It further depicts the movement of body, hands, fingers and other factors such as emotions on the face. The motion capture has been used to transmit the movements and for the sign language depiction. The dactyl-based modelling alphabet tech is developed using 3-d model. Having the efficiency on the Ukrainian sign language the realization is developed. This is based for the universal character and designed for further model. As times are improving and developing sign language emerges as one the best medium to communicate with the deaf community by having the supportive system to be in a help to society.

Nasri et al. [5] have proposed something great for the deaf community or hearing aid community by providing an app for the communication. But making an app for it is no simple task at it requires lot of efforts like memory utilization and a perfectly fined design to implement a such. What their application does is that they take a picture of a sign gesture and later converts is to a meaningful word. At first, they have compared the gesture using histogram that has been related to the sample test and moreover samples that are obliged to BRIEF to basically reduce the weight on the CPU and its time. They have explained a process on which on their app, it's very easy to add up a gesture and store it in their database for further and expand detection set. So lastly, they came strong with having an app as a translator instead of several applications that are being used lately by users.

Yang et al. [6] depict the usage of new and standardized model of communication system in which it basically targets the deaf people as that they have further explained. The system comprised of the following two scenarios like at first, Spanish

speeches to Spanish sign translation that makes the usage of a translator to break up the words and have them converted into the stream set of signs which in along with each other makes a sentence and also depicts in an avatar form. Now the

second scenario, the sign language that from a speech was generated then send to the generator, what this generator does that it makes out those signs into the Spanish word which will make sense which basically converts the sign language into the Spanish based spoken language words. Now whatever the data previously was generated is comprises of words, now the last step is to convert the words into a sentence that is word to speech conversion.

Wu et al. [7] applied a method of using a synthetic named animation making approach they have converted Malayalam language to Indian sign language. The intermediate representation in this method for sign language is being used by HamNoSys. In this method the application accepts some sets of words, say, either one or more and forms It in animated portion. There's an interactive system which further converts the portion of words in to HamNoSys designed structure. It's application pars everything that has been designed as it used by Kerala government to teach about sign language and subtle awareness.

Shou et al. [8] have built a system which works in a continuous manner in which the sign language gesture series is provided to make a automate training set and providing the spots sign from the set from training. They have proposed a system with instance learning as density matrix algorithm that supervises the sentence and figures out the compound sign gesture related to it with a supervision of noisy texts. The set at first that they had used to show the continuous data stream of words is further taken as a training set for recognizing the gesture posture. They have experimented this set on a confined set of automated data that is used for training of them, identification for them and detection stored a subtle sign data to them. It has been stored around thirty sign language data that was extracted from the designed proposal.

Miao et al. [9] the authors introduced two additional streams of cropped hands and mouth modalities. The full architecture was trained iteratively by frame-state alignments provided by the HMM. However, frame-state alignments can be noisy due

to the lack of frame-level ground truth annotations and such methods are forced to make strong initial assumptions on gloss boundaries in order to overcome HMM's limited learning capacity.

Lin et al. [10] the authors employed a 2D-CNN-LSTM architecture with CTC loss in parallel with a gloss-detection network to renew predictions. Later, they extended their work using temporal convolutions and a new iterative training scheme, achieving superior performance in CSLR datasets. However, the major weakness of CTC is the conditional independence assumption and therefore it fails to model intra-gloss dependencies.

CHAPTER 3

METHODOLOGY

3.1 SOFTWARE AND HARDWARE REQUIREMENTS:

3.1.1 *Software Requirements:*

- PyCharm
- Keras
- TensorFlow
- OpenCV

3.1.1.1 *PyCharm:*

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license. PyCharm is a hybrid-platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE! It supports two versions: v2.x and v3.x.

We can run PyCharm on Windows, Linux, or Mac OS. Additionally, it contains modules and packages that help programmers develop software using Python in less time and with minimal effort. Further, it can also be customized according to the requirements of developers.

3.1.1.2 *Keras:*

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Up until version 2.3 Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and Plaid ML. As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the Xception deep neural network model.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras is a high-level neural networks library written in python that works as a wrapper to TensorFlow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier.

3.1.1.3 TensorFlow:

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015. TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017.

While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions)

for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as *tensors*. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google. TensorFlow is an open source software library for numerical computation. First, we define the nodes of the computation graph, then inside a session, the actual computation takes place. TensorFlow is widely used in Machine Learning.

3.1.1.4 OpenCV:

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

Computer vision is a process by which we can understand the images and videos

how they are stored and how we can manipulate and retrieve data from them.

Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps. OpenCV (Open Source Computer Vision) is an open source library of programming functions used for real-time computer-vision. It is mainly used for image processing, video capture and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however bindings are available for Python, Java, MATLAB/OCTAVE.

3.1.2 Hardware Requirements:

- Windows 7 and above
- 1GB RAM
- Webcam
- Intel i3

3.2 OVERVIEW OF THE PLATFORM:

3.2.1 Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

It is used for:

- web development (server-side),
- software development,

- mathematics,
- system scripting.

3.2.2 Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.
- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

3.3 NEURAL NETWORKS:

A neural network is a computing model whose layered structure resembles the networked structure of neurons in the brain, with layers of connected nodes. A neural network can learn from data—so it can be trained to recognize patterns, classify data, and forecast future events. A neural network breaks down your input into layers of abstraction. It can be trained over many examples to recognize patterns in speech or images, for example, just as the human brain does.

Its behaviour is defined by the way its individual elements are connected and by the strength, or weights, of those connections. These weights are automatically adjusted during training according to a specified learning rule until the neural network performs the desired task correctly. Neural networks are especially well suited to perform pattern recognition to identify and classify objects or signals in speech, vision, and control systems. They can also be used for performing time- series prediction and modelling.

3.3.1 Advantages:

- NN have the ability to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.
- After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.
- Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables (like how they should be distributed). Additionally, many studies have shown that ANNs can better model heteroskedasticity i.e. data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed

relationships in the data. This is something very useful in financial time series forecasting (e.g. stock prices) where data volatility is very high.

3.3.2 Applications:

Because of their ability to reproduce and model nonlinear processes, Artificial neural networks have found applications in many disciplines. Application areas include system identification and control (vehicle control, trajectory prediction, process control, natural resource management), quantum chemistry, general game playing, pattern recognition (radar systems, face identification, signal classification, 3D reconstruction, object recognition and more), sequence recognition (gesture, speech, handwritten and printed text recognition), medical diagnosis, finance (e.g. automated trading systems), data mining, visualization, machine translation, social network filtering and e-mail spam filtering. ANNs have been used to diagnose several types of cancers and to distinguish highly invasive cancer cell lines from less invasive lines using only cell shape information.

ANNs have been used to accelerate reliability analysis of infrastructures subject to natural disasters and to predict foundation settlements. ANNs have also been used for building black-box models in geoscience: hydrology, ocean modelling and coastal engineering, and geomorphology. ANNs have been employed in cybersecurity, with the objective to discriminate between legitimate activities and malicious ones. For example, machine learning has been used for classifying Android malware, for identifying domains belonging to threat actors and for detecting URLs posing a security risk. Research is underway on ANN systems designed for penetration testing, for detecting botnets, credit cards frauds and network intrusions.

ANNs have been proposed as a tool to solve partial differential equations in physics and simulate the properties of many-body open quantum systems. In brain research ANNs have studied short-term behaviour of individual neurons, the dynamics of neural circuitry arise from interactions between individual neurons and how behaviour can arise from abstract neural modules that represent complete subsystems. Studies considered long-and short-term plasticity of neural systems and their relation to learning and memory from the individual neuron to the system level.

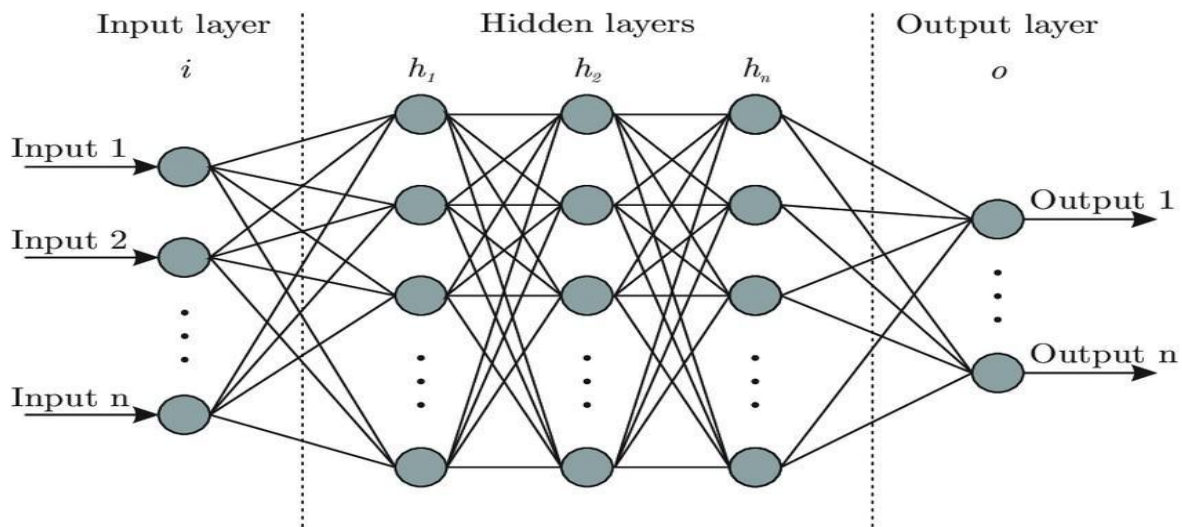


Fig 3.1 Neural network Architecture

3.4 CONVOLUTIONAL NEURAL NETWORKS:

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

CNNs are regularized versions of multilayer perceptron's. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include varying the weights as the loss function gets minimized while randomly trimming connectivity. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in the filters. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex.

Individual cortical neurons respond to stimuli only in a

restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters or convolution kernels that in traditional algorithms are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

CNN help in running neural networks directly on images and are more efficient and accurate than many of the deep neural networks. ConvNet models are easy and faster to train on images comparatively to the other models. One of the limitations of the CNN model is that they cannot be trained on a different dimension of images. So, it is mandatory to have same dimension images in the dataset. We'll check the dimension of all the images of the dataset so that we can process the images into having similar dimensions. In this dataset, the images have a very dynamic range of dimensions from $16 \times 16 \times 3$ to $128 \times 128 \times 3$ hence cannot be passed directly to the ConvNet model. We need to compress or interpolate the images to a single dimension. Not, to compress much of the data and not to stretch the image too much we need to decide the dimension which is in between and keep the image data mostly accurate. I've decided to use dimension $64 \times 64 \times 3$.

3.4.1 Layers of CNN:

1. **Convolution Layer:** In convolution layer we take a small window size [typically, of length 5×5] that extends to the depth of the input matrix. The layer consist of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some colour.

2. **Pooling Layer:** We use pooling layer to decrease the size of activation matrix

and ultimately reduce the learnable parameters. There are two type of pooling:

a) **Max Pooling:** In max pooling we take a window size [for example window of size 2×2], and only take the maximum of 4 values. We'll slide this window and continue this process, so we'll finally get an activation matrix half of its original size.

b) **Average Pooling:** In average pooling we take the average of all values in a window.

3. **Fully Connected Layer:** In convolution layer neurons are connected only to a local region, while in a fully connected region, we'll connect all the inputs to neurons.

4. **Final Output Layer:** After getting values from the fully connected layer, we'll connect them to the final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.

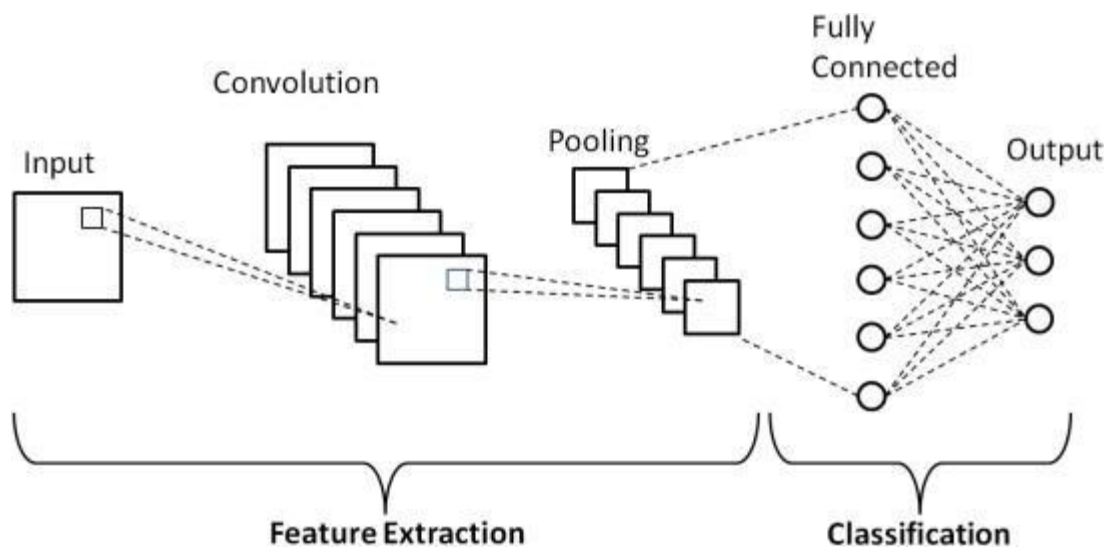


Fig 3.2 CNN Architecture

3.5 PROPOSED SYSTEM:

For interaction between normal people and D&M people a language barrier is created as sign language structure which is different from normal text. So, they depend on vision-based communication for interaction. If there is a common interface that converts the sign language to text the gestures can be easily understood by the other people. So, research has been made for a vision-based interface system where D&M people can enjoy communication without really knowing each other's language. The aim is to develop a user-friendly human computer interfaces (HCI) where the

computer understands the human sign language. There are various sign languages all over the world, namely American

Sign Language (ASL), French Sign Language, British Sign Language (BSL), Indian Sign language, Japanese Sign Language and work has been done on other languages all around the world.

3.6 SYSTEM ARCHITECTURE:

3.6.1 Module Description:

1. Image Acquisition:

The gestures are captured through the web camera. This OpenCV video stream is used to capture the entire signing duration. The frames are extracted from the stream and are processed as grayscale images with the dimension of 50*50. This dimension is consistent throughout the project as the entire dataset is sized exactly the same.

2. Hand Region Segmentation & Hand Detection and Tracking:

The captured images are scanned for hand gestures. This is a part of pre- processing before the image is fed to the model to obtain the prediction. The segments containing gestures are made more pronounced. This increases the chances of prediction by many folds.

3. Hand Posture Recognition:

The pre-processed images are fed to the keras CNN model. The model that has already been trained generates the predicted label. All the gesture labels are assigned with a probability. The label with the highest probability is treated to be the predicted label.

4. Display as Text & Speech:

The model accumulates the recognized gesture to words. The recognized words are converted into the corresponding speech using the pyttsx3 library. The text to speech result is a simple work around but is an invaluable feature as it gives a feel of an actual verbal conversation.

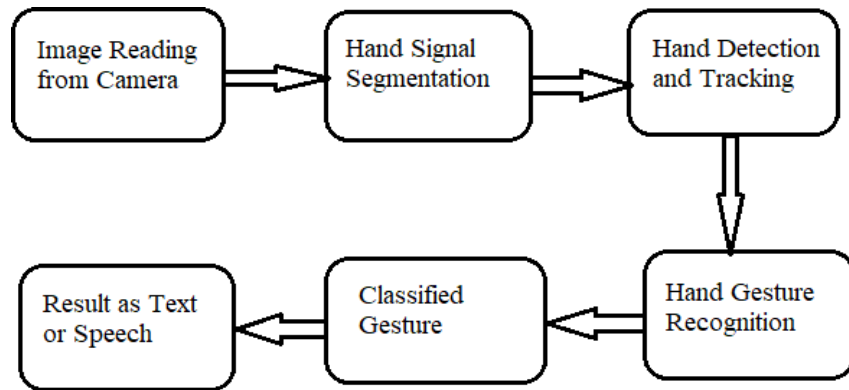


Fig 3.3 System Architecture

3.7 DATASET GENERATION:

For the project we tried to find already made datasets but we couldn't find dataset in the form of raw images that matched our requirements. All we could find were the datasets in the form of RGB values. Hence, we decided to create our own data set. Steps we followed to create our data set are as follows. We used Open computer vision (OpenCV) library in order to produce our dataset. Firstly, we captured around 800 images of each of the symbol in ASL for training purposes and around 200 images per symbol for testing purpose.

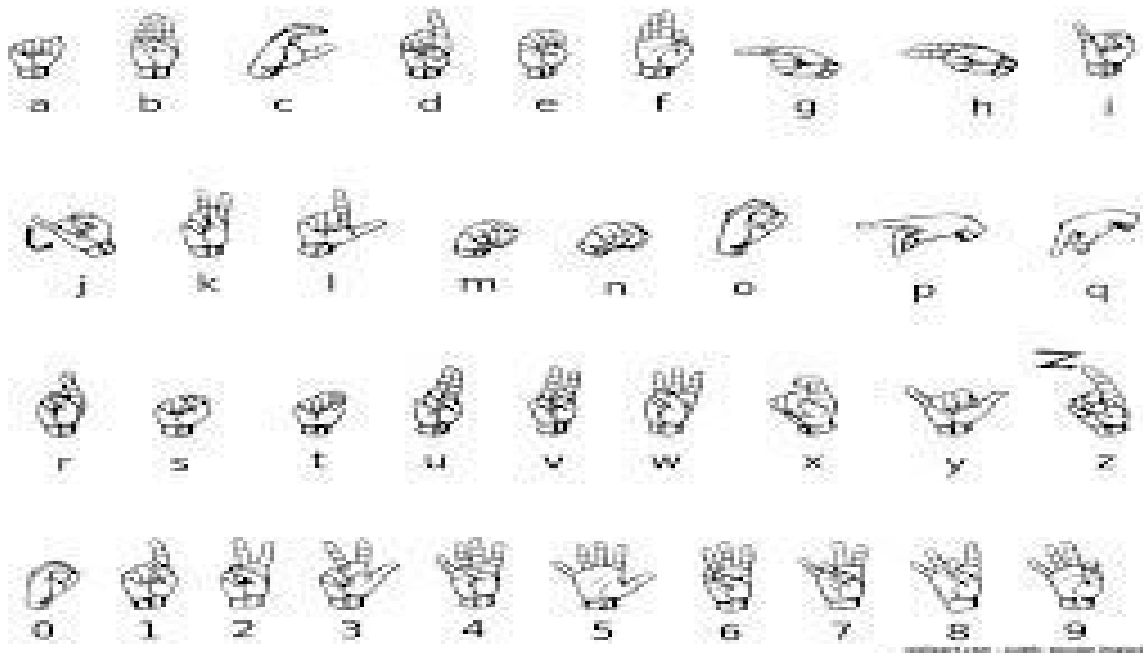


Fig 3.4 Dataset

3.8 GESTURE CLASSIFICATION:

Algorithm Layer 1:

1. Apply gaussian blur filter and threshold to the frame taken with OpenCV to get the processed image after feature extraction.
2. This processed image is passed to the CNN model for prediction and if a letter is detected for more than 50 frames then the letter is printed and taken into consideration for forming the word.
3. Space between the words are considered using the blank symbol.

Algorithm Layer 2:

1. We detect various sets of symbols which show similar results on getting detected.
2. We then classify between those sets using classifiers made for those sets only.

3.8.1 Layer 1:

CNN Model:

1. **1st Convolution Layer:** The input picture has resolution of 128x128 pixels. It is first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 126X126 pixel image, one for each Filter-weights.
2. **1st Pooling Layer:** The pictures are down sampled using max pooling of 2x2 i.e. we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels.
3. **2nd Convolution Layer:** Now, these 63 x 63 from the output of the first pooling layer is served as an input to the second convolutional layer. It is processed in the second convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 60 x 60-pixel image.
4. **2nd Pooling Layer:** The resulting images are down sampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.
5. **1st Densely Connected Layer:** Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of $30 \times 30 \times 32 = 28800$ values. The input to this layer is an array of 28800 values. The output of these layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.
6. **2nd Densely Connected Layer:** Now the output from the 1st Densely Connected Layer are used as an input to a fully connected layer with 96 neurons.
7. **Final layer:** The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

3.8.2 Layer 2:

We are using two layers of algorithms to verify and predict symbols which are more similar to each other so that we can get as close as we can get to detect the symbol

shown. In our testing we found that following symbols were not showing properly and were giving other symbols also:

1. For D: R and U
2. For U: D and R
3. For I: T, D, K and I
4. For S: M and N

So, to handle above cases we made three different classifiers for classifying these sets:

1. {D, R, U}
2. {T, K, D, I}
3. {S, M, N}

3.9 IMPLEMENTATION:

1. Whenever the count of a letter detected exceeds a specific value and no other letter is close to it by a threshold, we print the letter and add it to the current string (In our code we kept the value as 50 and difference threshold as 20).
2. Otherwise we clear the current dictionary which has the count of detections of present symbol to avoid the probability of a wrong letter getting predicted.
3. Whenever the count of a blank (plain background) detected exceeds a specific value and if the current buffer is empty no spaces are detected.
4. In other case it predicts the end of word by printing a space and the current gets appended to the sentence.

3.9.1 AutoCorrect Features:

A python library Hunspell_suggest is used to suggest correct alternatives for each (incorrect) input word and we display a set of words matching the current word in

which the user can select a word to append it to the current sentence. This helps in reducing mistakes committed in spellings and assists in predicting complex words.

3.9.2 Training and Testing:

We convert our input images (RGB) into grayscale and apply gaussian blur to remove unnecessary noise. We apply adaptive threshold to extract our hand from the background and resize our images to 128 x 128. We feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above. The prediction layer estimates how likely the image will fall under one of the classes. So, the output is normalized between 0 and 1 and such that the sum of each values in each class sums to 1. We have achieved this using SoftMax function.

At first the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labelled data. The cross entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labelled value and is zero exactly when it is equal to the labelled value. Therefore, we optimized the cross- entropy by minimizing it as close to zero. To do this in our network layer we adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy. As we have found out the cross-entropy function, we have optimized it using Gradient Descent in fact with the best gradient descent optimizer is called Adam Optimizer.

CHAPTER 4 RESULTS AND DISCUSSIONS

4.1 RESULTS:

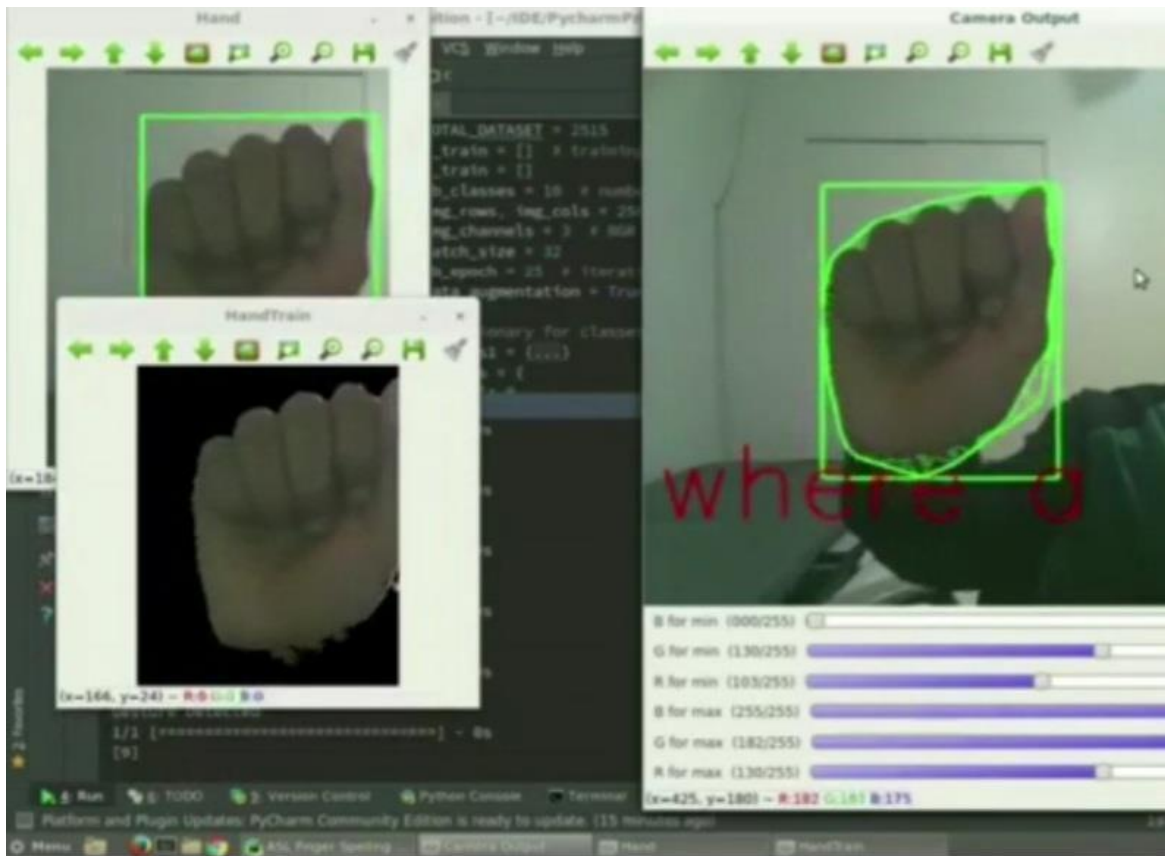
We have achieved an accuracy of 95.8% in our model using only layer 1 of our algorithm, and using the combination of layer 1 and layer 2 we achieve an accuracy of 98.0%, which is a better accuracy than most of the current research papers on American sign language. Most of the research papers focus on using devices like Kinect for hand detection. In they build a recognition system for Flemish sign language using convolutional neural networks and Kinect and achieve an error rate of 2.5%.

In a recognition model is built using hidden markov model classifier and a vocabulary of 30 words and they achieve an error rate of 10.90%. They also used CNN for their recognition system. One thing should be noted that our model doesn't uses any background subtraction algorithm whiles some of the models present above do that. So, once we try to implement background subtraction in our project the accuracies may vary. On the other hand, most of the above projects use Kinect devices but our main aim was to create a project which can be used with readily available resources. A sensor like Kinect not only isn't readily available but also is expensive for most of audience to buy and our model uses a normal webcam of the laptop hence it is great plus point.

Sign language recognition includes two main categories, which are isolated sign language recognition and continuous sign language recognition. The supervision information is a key difference between the two categories. While isolated sign language recognition is similar to the action recognition area, the continuous sign language recognition concerns about not only the recognition task but also the accurate alignment between the input video segments and the corresponding sentence-level labels. Generally, continuous sign language recognition is more challenging than isolated sign language recognition. Indeed, isolated sign language recognition can be considered as a subset of continuous sign language recognition.

Two factors play a key role in the performance evaluation of continuous sign language recognition, which include feature extraction from frame sequences of the

input video and alignment between the features of each video segment and the corresponding sign label.



This project is totally based on ASL finger spelling. The above image shows that the CNN model captures the hand gestures which was made in front of camera. There are totally three screens displayed in the image. The screen to the right shows the full image where the text is also displayed. Next screen which is to the corner right shows the cropped image of the hand and the final screen shows the cropped image which eliminates the background. The captured image is converted into text as shown.

4.2 DISCUSSIONS:

We will compare the effects of the proposed depth motion features and the classical RGB optical_flow features in this section. When the depth image quality is satisfactory, the depth motion feature shows excellent performance, especially for the behaviour

series with apparent displacements in distance to the camera. We

show the example of qualitative results on IsoGD in that the depth motion feature obtains a clearer and more complete motion trajectory than the RGB optical flow, making the feature more vivid, when the motion has obvious depth displacement and the horizontal displacement is small displays that when the moving parts are similar in colour, the optical flow features are obvious poor. At the same time, the RGB optical stream cannot locate the pixels through the colour block, but the depth motion network can extract the motion features through the change of the depth information. In addition, depth information ignores the effects of light and shadow changes, which greatly avoids the interference of the environment and captures the gesture features more accurate. The fatal effect of depth image quality on feature extraction. The depth data of IsoGD has some depth missing area, which I represented by a depth value of zero. The occlusion problem and the surface material of the object will affect the acquisition of the Kinect depth image. These noises affect the quality of the depth image seriously, resulting in inaccurate depth motion feature extraction. If the background noise is filtered in pre-process, the quality of depth motion features will be significantly improved. We segmented foreground character on CSL by the contour extraction algorithm Morphological, as illustrated in Contour extraction algorithm can effectively remove the interference of background noise, but it performs poorly on the IsoGD dataset with complex background. Choosing the right pre-processing method to optimize the data set usually yields better results on IsoGD.

Obtaining more descriptive and discriminative features from the video frames could result in a better performance for a continuous sign language recognition system. While recent models in continuous sign language recognition have a rising trend in model performance relying on deep learning capabilities in computer vision and NLP, there is still much room for performance improvement in this area. Considering the attention mechanism, using multiple input modalities to benefit from multi-channel information, learning structured spatio-temporal patterns (such as Graph Neural Networks models), and employing the prior knowledge on sign language are only some of the possible future directions in this area.

CHAPTER 5

CONCLUSIONS AND FUTURE SCOPE

5.1 CONCLUSIONS:

The project is a simple demonstration of how CNN can be used to solve computer vision problems with an extremely high degree of accuracy. A finger spelling sign language translator is obtained which has an accuracy of 95%. The project can be extended to other sign languages by building the corresponding dataset and training the CNN. Sign languages are spoken more in context rather than as finger spelling languages, thus, the project is able to solve a subset of the Sign Language translation problem. The main objective has been achieved, that is, the need for an interpreter has been eliminated. There are a few finer points that need to be considered when we are running the project. The thresh needs to be monitored so that we don't get distorted grayscales in the frames. If this issue is encountered, we need to either reset the histogram or look for places with suitable lighting conditions. We could also use gloves to eliminate the problem of varying skin complexion of the signee. In this project, we could achieve accurate prediction once we started testing using a glove. The other issue that people might face is regarding their proficiency in knowing the ASL gestures. Bad gesture postures will not yield correct prediction. This project can be enhanced in a few ways in the future, it could be built as a web or a mobile application for the users to conveniently access the project, also, the existing project only works for ASL, it can be extended to work for other native sign languages with enough dataset and training. This project implements a finger spelling translator, however, sign languages are also spoken in a contextual basis where each gesture could represent an object, verb, so, identifying this kind of a contextual signing would require a higher degree of processing and natural language processing (NLP). This is beyond the scope of this project.

For aligned multimodal input, the ASL approach covers key information and effectively removes redundancy. Local focus of the hand optimizes the input of spatial network. And D-shift Net generates depth motion features to explore depth

information effectively. A convolutional fusion is subsequently conducted to fuse two-stream features and better recognition results. Our future work could involve

optimizing the image quality of depth video for more effective motion features extraction and uniting both depth motion features and RGB optical flow, as well as improving the recognition speed without reducing precision.

5.2 SUMMARY:

Based on state-of-the-art methods, this article clearly depicts the vital role of machine learning methods in automatic recognition of sign languages, and addresses the need of subunit sign modelling for continuous sign language. This article mainly concentrates on solving three major issues in SLR namely extraction and selection of robust subunit features, handling epenthesis movements and implementing the framework for subunit sign modelling.

The first research frontier at the feature level, considers the problem that makes hand segmentation and grouping hard which includes using short sleeves, signing in complex background and interaction without external interface devices. The second frontier at the sentence level considers the problem of handling epenthesis movements that are pruning epenthesis movements and classifying sign gestures. The final contribution of the research frontier is to introduce the novel subunit sign modelling and signer adaptation framework for automatic sign language recognition system to recognize large vocabulary which involves subunit extraction, subunit sign lexicon construction, subunit sharing and sign classification.

5.3 FUTURE SCOPES:

We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. We are also thinking of improving the pre-processing to predict gestures in low light conditions with a higher accuracy.

In future work, proposed system can be developed and implemented using Raspberry Pi. Image Processing part should be improved so that System would be able to communicate in both directions i.e. it should be capable of converting normal

language to sign language and vice versa. We will try to recognize signs which include motion. Moreover, we will focus on converting the sequence of gestures into

text i.e. word and sentences and then converting it into the speech which can be heard.

We have successfully designed and implemented a sign language interpretation system for Indian Sign language with the help of a wearable hand glove. This device allows translation of single-handed signs using microtouch switch and Arduino. The gestures performed by the user are converted into text and speech with the help of input matrix assigned in the microtouch sensor, this can be easily understood by the normal people. Also, in this proposed work the reverse operation was carried out by using the mobile application. (i.e.) A Recognized speech from the normal person to text service for the user (hearing impaired) to understand what the other person is speaking; this was performed by an Android based mobile application. Hence this device allows two-way communications. This kind of translator device provides a social compatibility to the user who is both speech and aurally challenged. The system can be improved by allowing multi-language to be displayed and converted to speech. Furthermore, other sensors (accelerometers, capacitive flex sensors etc.,) can be integrated with the system for recognition of movement of the hand such as swapping, rotation, tilting etc. Mobile applications can be developed for replacing the LCD display and the speaker which minimizes the hardware.

REFERENCES

- [1] C. Sun, T. Zhang, B. K. Bao, C. Xu and T. Mei, "Discriminative exemplar coding for sign language recognition with kinect", IEEE Transactions on Cybernetics, vol. 43, no. 5, pp. 1418-1428, 2013.
- [2] W. C. Hall, "What You Don't Know Can Hurt You: The Risk of Language Deprivation by Impairing Sign Language Development in Deaf Children", Maternal and Child Health Journal, pp. 1-5, 2017.
- [3] R. C. Dalawis, K. D. R. Olayao, E. G. I. Ramos and M. J. C. Samonte, "Kinect-Based Sign Language Recognition of Static and Dynamic Hand Movements", Eighth International Conference on Graphic and Image Processing, pp. 1022501-1022501.
- [4] Suharjito Ricky, Anderson Fanny, Wiryana Meita, Chandra Ariesta and Gede Putra Kusuma, "Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output", 2nd International Conference on Computer Science and Computational Intelligence 2017 ICCSCI 2017, 3-14 October 2017.
- [5] A. M. Olson and L. Swabey, "Communication Access for Deaf People in Healthcare Settings: Understanding the Work of American Sign Language Interpreters", Journal for healthcare quality: official publication of the National Association for Healthcare Quality, 2016.
- [6] G. Anantha Rao, K. Syamala, P.V.V. Kishore and A.S.C.S. Sastry, Deep Convolutional Neural Networks for Sign Language Recognition.
- [7] Vannesa Mueller, Amanda Sepulveda and Sarai Rodriguez, The effects of baby sign training on child development In:Early Child Development and Care, vol. 184, no. 8, pp. 1178-1191, 2014.
- [8] F. S. Chen, C. M. Fu and C. L. Huang, "Hand gesture recognition using a real-time tracking method and hidden Markov models", Image and vision computing, vol. 21, no. 8, pp. 745-758, 2003.
- [9] T. Yang, Y. Xu, and "A. , Hidden Markov Model for Gesture Recognition", CMU-RI-TR-94 10, Robotics Institute, Carnegie Mellon Univ.,Pittsburgh,PA, May 1994.
- [10] Pujan Ziaie, Thomas M iller, Mary Ellen Foster, and Alois Knoll"A Na ive Bayes Munich, Dept. of Informatics VI, Robotics and Embedded Systems, Boltzmannstr. 3,

DE-85748 Garching, Germany.

APPENDIX

A. SOURCE CODE:

```
import cv2
import numpy
import math
import os

from keras.preprocessing.image import img_to_array, load_img
from keras.optimizers import SGD
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
from PIL import Image
import keras

# path to the dataset
paths = ['/home/snrao/IDE/PycharmProjects/ASL_Finger_Spelling
Recognition/asl_dataset']
TOTAL_DATASET = 2515
x_train = [] # training lists y_train =
[]
nb_classes = 36 # number of classes
img_rows, img_cols = 400, 400 # size of training images
img_channels = 3 # BGR channels
batch_size = 32
nb_epoch = 100 # iterations for training
data_augmentation = True

# dictionary for classes from char to numbers
classes = {
    '0': 0,
    '1': 1,
```

'2': 2,

```
}
```

```
# load the dataset and populate xtrain and ytrain
```

```
def load_data_set():
```

```
    for path in paths:
```

```
        for root, directories, filenames in os.walk(path):
```

```
            for filename in filenames:
```

```
                if filename.endswith(".jpeg"):
```

```
                    fullpath = os.path.join(root, filename)
```

```
                    img = load_img(fullpath)
```

```
                    img = img_to_array(img)
```

```
                    x_train.append(img)
```

```
                    t = fullpath.rindex('/')
```

```
                    fullpath = fullpath[0:t]
```

```
                    n = fullpath.rindex('/')
```

```
                    y_train.append(classes[fullpath[n + 1:t]])
```

```
# create a model for training and return the model
```

```
def make_network(x_train):
```

```
    model = Sequential()
```

```
    model.add(Convolution2D(32, 3, 3, border_mode='same',  
                            input_shape=x_train.shape[1:]))
```

```
    model.add(Activation('relu'))
```

```
    model.add(Convolution2D(32, 3, 3))
```

```
    model.add(Activation('relu'))
```

```
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    model.add(Convolution2D(64, 3, 3, border_mode='same'))
```

```
model.add(Activation('relu')) model.add(Convolution2D(64, 3,  
3)) model.add(Activation('relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
```

```
model.add(Dense(512))
```

```
model.add(Activation('relu'))
```

```
model.add(Dense(nb_classes))
```

```
model.add(Activation('softmax'))
```

```
return model
```

```
# training model which was created
```

```
def train_model(model, X_train, Y_train):
```

```
    # let's train the model using SGD + momentum (how original). sgd =
```

```
    SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
```

```
    model.compile(loss='categorical_crossentropy',
```

```
                  optimizer=sgd,
```

```
                  metrics=['accuracy'])
```

```
    model.fit(X_train, Y_train,
```

```
             batch_size=batch_size,
```

```
             nb_epoch=nb_epoch)
```

```
# loads data set, converts the training arrays into required formats of numpy arrays  
and calls make_network to
```

```
# create a model and then calls train_model to train it and then saves the model in  
disk. OR just loads the model
```

```
# from disk. def
```

```
trainData():
```

```
    load_data_set()
```

```
a = numpy.asarray(y_train) y_train_new =  
a.reshape(a.shape[0], 1)
```

```

X_train = numpy.asarray(x_train).astype('float32')
X_train = X_train / 255.0
Y_train = np_utils.to_categorical(y_train_new, nb_classes)

# run this if model is not saved.
model = make_network(numpy.asarray(x_train))
train_model(model,X_train,Y_train)
model.save('/home/snrao/IDE/PycharmProjects/ASL      Finger      Spelling
Recognition/keras.model')

# run this if model is already saved on disk.
# model = keras.models.load_model('/home/snrao/IDE/PycharmProjects/ASL
Finger Spelling Recognition/keras.model')

return model

model = trainData()

# called from main, when gesture is recognized. The gesture image is cropped and
sent to this function.
# noinspection PyInterpreter
def identifyGesture(handTrainImage): #
    saving the sent image for checking
    # cv2.imwrite("/home/snrao/IDE/PycharmProjects/ASL      Finger      Spelling
Recognition/a0.jpeg", handTrainImage)

    # converting the image to same resolution as training data by padding to reach 1:1
aspect ration and then
    # resizing to 400 x 400. Same is done with training data in preprocess_image.py.
    Opencv image is first

```

converted to Pillow image to do this.

```
handTrainImage = cv2.cvtColor(handTrainImage, cv2.COLOR_BGR2RGB)
```

```

img = Image.fromarray(handTrainImage)
img_w, img_h = img.size
M = max(img_w, img_h)
background = Image.new('RGB', (M, M), (0, 0, 0))
bg_w, bg_h = background.size
offset = ((bg_w - img_w) / 2, (bg_h - img_h) / 2)
background.paste(img, offset)
size = 400,400
background = background.resize(size, Image.ANTIALIAS)

# saving the processed image for checking.
# background.save("/home/snrao/IDE/PycharmProjects/ASL Finger Spelling
Recognition/a.jpeg")

# get image as numpy array and predict using model
open_cv_image = numpy.array(background)
background = open_cv_image.astype('float32')
background = background / 255
background = background.reshape((1,) + background.shape)
predictions = model.predict_classes(background)

# print predicted class and get the class name (character name) for the given class
number and return it
print predictions
key = (key for key, value in classes.items() if value == predictions[0]).next()
return key

import os
import numpy
from PIL import Image
import cv2

```

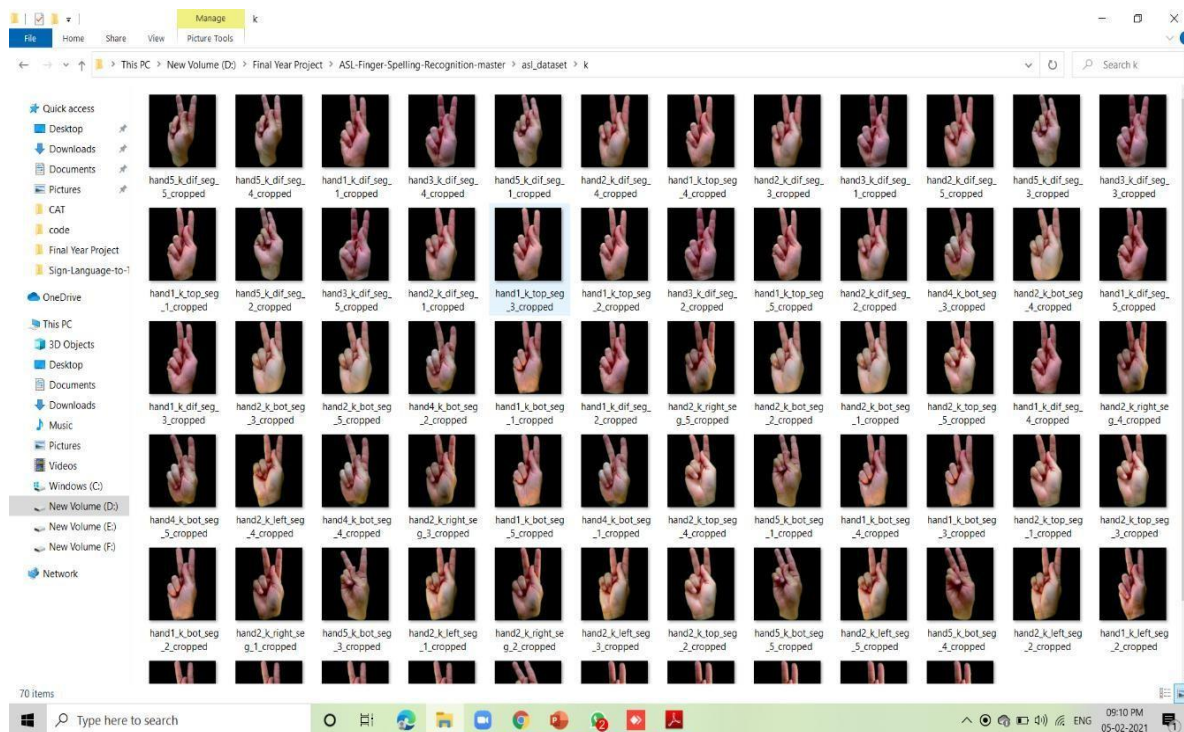
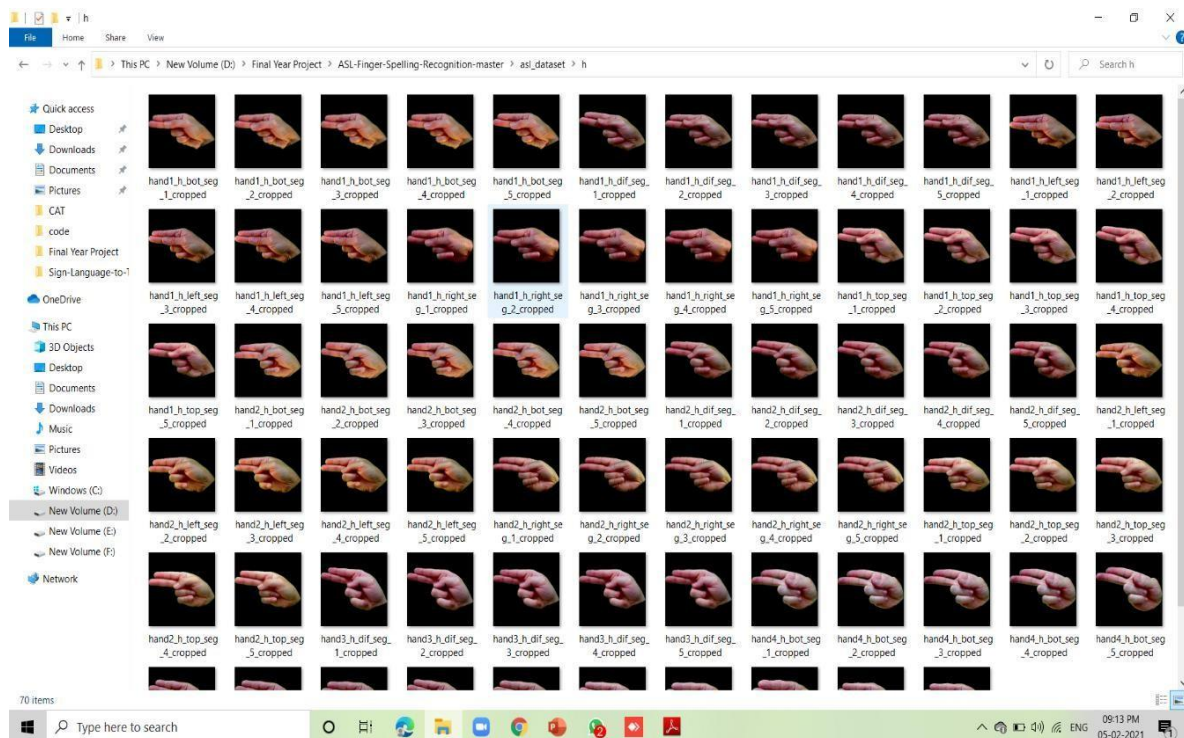
```
def getDimensions(filename):
```

```
img = cv2.imread(filename)
height, width, channel = img.shape
return height, width
```

```
paths = ['C:\Users\Ashish\Work\HAR\Mid Term Project']
dimensions = []
for path in paths:
    for root, directories, filenames in os.walk(path):
        for filename in filenames:
            if filename.endswith(".png"):
                l, w = getDimensions(os.path.join(root, filename))
                dimensions.append([l, w])
```

```
A = numpy.asarray(dimensions)
L, W = A.max(axis=0)
print L, W
```

B. SCREENSHOTS:



The above screen shots show few of the 7821 datasets which we got from an open source. Each alphabet has many images according to skin colour, lighting and so on.

```
ASL Finger Spelling Recognition - [~/IDE/PycharmProjects/ASL Finger Spelling Recognition] - .../main.py - PyCharm Community Edition 2016.2.2
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
```

```
ASL Finger Spelling Recognition main.py
```

```
Project
```

- ASL Finger Spelling Recognition
 - asl_dataset
 - dataset
 - a.jpg
 - keras.model
 - keras1.model
 - main.py
 - palm.xml
 - preprocess_image.py
 - trial.py
 - External Libraries

```
main.py
```

```
24 batch_size = 32
```

```
25 nb_epoch = 25 # iterations for training
```

```
26 data_augmentation = True
```

```
27
```

```
28 # dictionary for classes from char to numbers
```

```
29 classes1 = {...}
```

```
67 classes = {...}
```

```
68
```

```
69 # load the dataset and populate xtrain and ytrain
```

```
81
```

```
82 def load_data_set():
```

```
83     for path in paths:
```

```
84         for root, directories, filenames in os.walk(path):
```

```
85             for filename in filenames:
```

```
86
```

```
Run main
```

```
/usr/bin/python2.7 "/home/snrao/IDE/PycharmProjects/ASL Finger Spelling Recognition/main.py"
```

```
Using TensorFlow backend.
```

```
Epoch 1/25 |
```

```
448/760 [=====>.....] - ETA: 156s - loss: 2.3834 - acc: 0.1849
```

```
Run TODO Version Control Python Console Terminal
```

```
Platform and Plugin Updates: PyCharm Community Edition is ready to update. (14 minutes ago)
```

```
4.82 LFs UTF-8: Git: GestureRecognitionBranch: 16:13
```

```
ASL Finger Spelling Recognition - [~/IDE/PycharmProjects/ASL Finger Spelling Recognition] - .../main.py - PyCharm Community Edition 2016.2.2
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
```

```
ASL Finger Spelling Recognition main.py
```

```
Project
```

- ASL Finger Spelling Recognition
 - asl_dataset
 - dataset
 - a.jpg
 - keras.model
 - keras1.model
 - main.py
 - palm.xml
 - preprocess_image.py
 - trial.py
 - External Libraries

```
main.py
```

```
343 # hist2 = cv2.calcHist(prevHandImage, [0, 1, 2], None, [0, 0, 0],
```

```
344 # (0, 256, 0, 256, 0, 256))
```

```
345 # hist2 = cv2.normalize(hist2, hist2).flatten()
```

```
346 # d = cv2.compareHist(hist1, hist2, cv2.HISTCMP_CHISQR)
```

```
347 # if d<0.9:
```

```
348 # print(d)
```

```
349 # prevHandImage = handImage
```

```
350
```

```
351 # haar cascade classifier to detect palm and gestures. Not very accurate though.
```

```
352 # Needs more training to become accurate.
```

```
353 gray = cv2.cvtColor(handImage, cv2.COLOR_BGR2HSV)
```

```
354 palm = palm_cascade.detectMultiScale(gray)
```

```
355 for (x, y, w, h) in palm:
```

```
356     cv2.rectangle(sourceImage, (x, y), (x + w, y + h), (255, 0, 0), 2)
```

```
357     # roi_gray = gray[y:y + h, x:x + w]
```

```
358     roi_color = sourceImage[y:y + h, x:x + w]
```

```
359
```

```
360 # to show convex hull in the image
```

```
361 hull = cv2.convexHull(cnt, returnPoints=False)
```

```
362 defects = cv2.convexityDefects(cnt, hull)
```

```
363
```

```
364 # counting defects in convex hull. To find center of palm. Center is average of defect points.
```

```
365 count_defects = 0
```

```
366 for i in range(defects.shape[0]):
```

```
367     s, e, f, d = defects[i, 0]
```

```
368     start = tuple(cnt[s][0])
```

```
369     end = tuple(cnt[e][0])
```

```
370     far = tuple(cnt[f][0])
```

```
371     if count_defects == 0:
```

```
372         center_of_palm = far
```

```
373     a = math.sqrt((end[0] - start[0]) ** 2 + (end[1] - start[1]) ** 2)
```

```
374     b = math.sqrt((far[0] - start[0]) ** 2 + (far[1] - start[1]) ** 2)
```

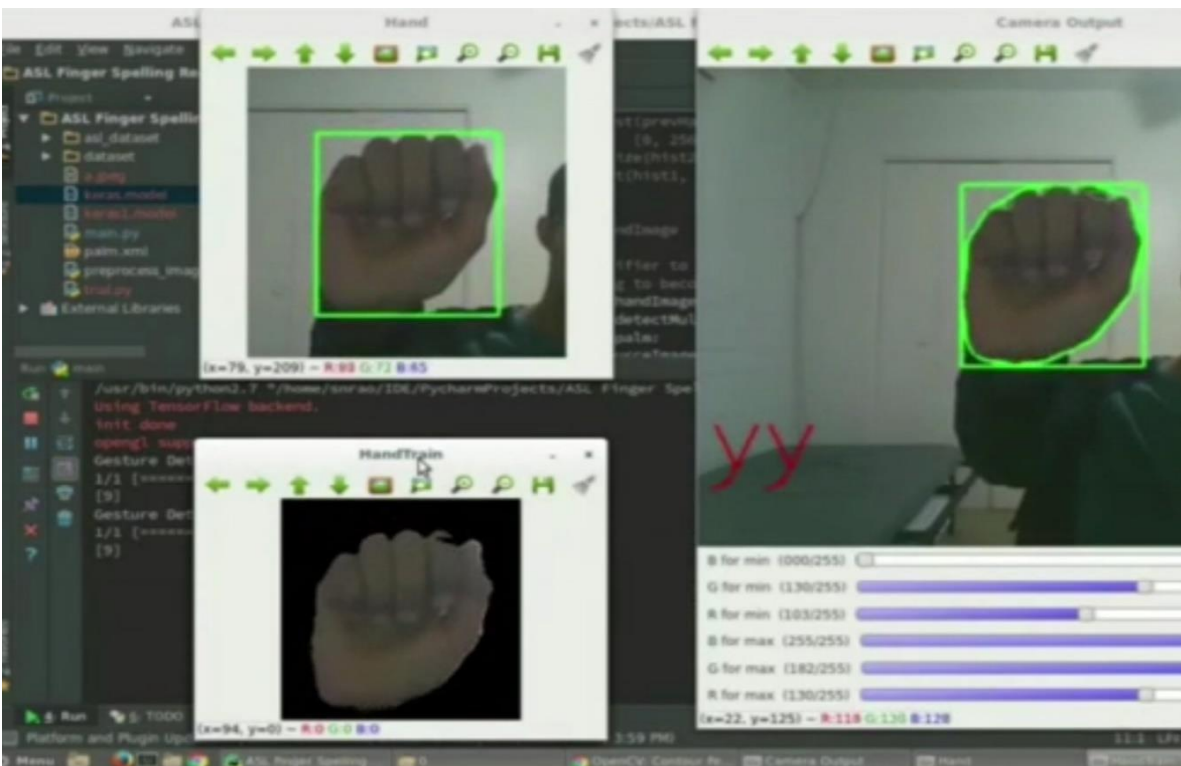
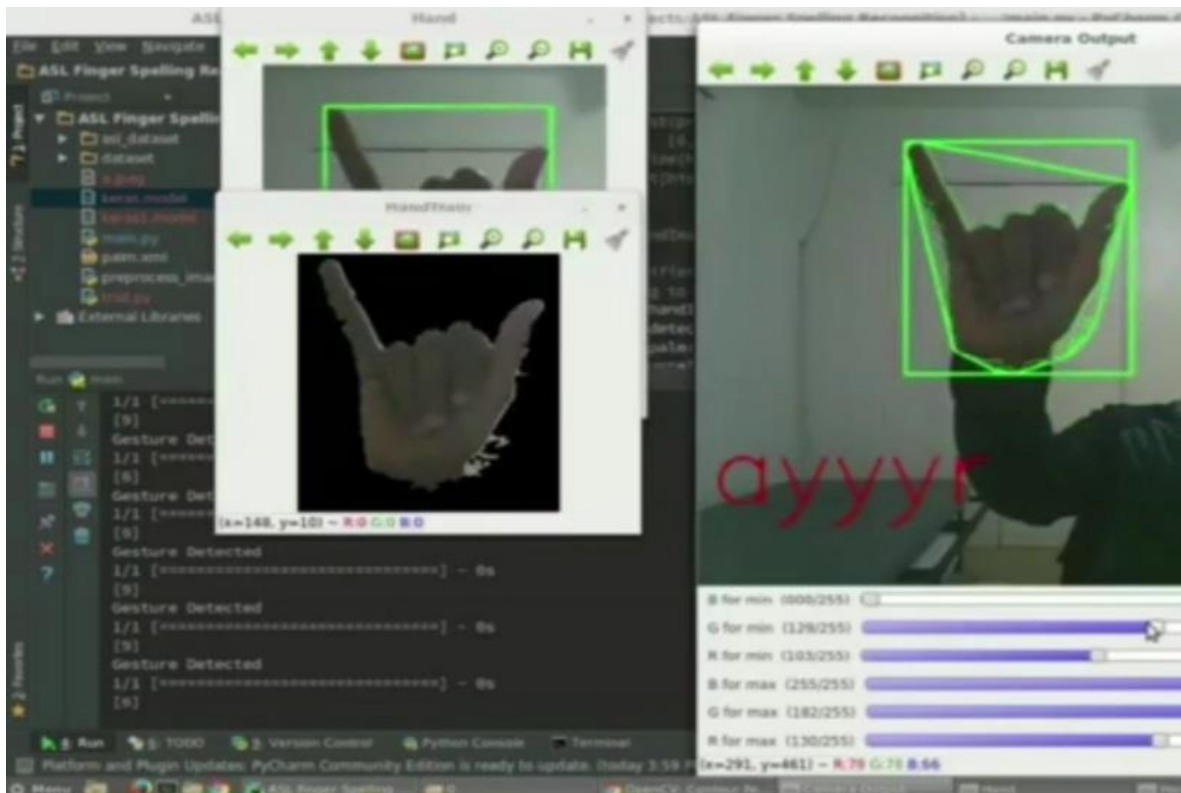
```
375     c = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
```

```
Run TODO Version Control Python Console Terminal
```

```
Platform and Plugin Updates: PyCharm Community Edition is ready to update. (today 3:59 PM)
```

```
7:1 LFs UTF-8: Git: GestureRecognitionBranch: 17:15
```

The above two screenshots show the main code for training the CNN model. The training of the CNN model is the main module of the project. The screenshot also shows the implementation of the code.



The above two screenshots show the resultant images where the finger spelling sign language is translated to text as shown.

C. PLAGIARISM REPORT:

Final paper			
ORIGINALITY REPORT			
13%	10%	4%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	www.ijert.org Internet Source	5%	
2	pydeeplearning.weebly.com Internet Source	2%	
3	Submitted to Visvesvaraya Technological University, Belagavi Student Paper	1%	
4	Submitted to Segi University College Student Paper	1%	
5	Submitted to Liverpool John Moores University Student Paper	1%	
6	www.ijircce.com Internet Source	1%	
7	Submitted to San Joaquin Delta Community College Student Paper	1%	
8	Imran Shafiq Ahmad, Boubakeur Boufama. "Automatic Vehicle Identification Through Visual Features", Proceedings of the 17th International	<1%	
Conference on Advances in Mobile Computing & Multimedia, 2019 Publication			
9	ib.compscihub.net Internet Source	<1%	
10	www.turkaramamotoru.com Internet Source	<1%	
11	pdfs.semanticscholar.org Internet Source	<1%	