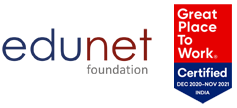
**IBM SkillsBuild Program**

###### AI &ML Internship





Sign Language to Text Conversion for Dumb and Deaf

**by**

**Sachin Kumar Singh- STU6079bc91184c41618590865**

## **Under the supervision of**

### **Bassar Patel**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Contents** | **Page No.** |
| **1** | Abstract | 02 |
| **2** | Introduction | 03 |
| **3** | Motivation | 06 |
| **4** | Literature Review | 07 |
| **5** | Keyword and Definition | 08 |
| **6** | Methodologies | 12 |
| **7** | Challenges faced | 13 |
| **8** | Results | 14 |
| **9** | Conclusion | 14 |
| **10** | Future Scope | 15 |
| **11** | References | 15 |
| **12** | Appendix | 17 |

Table of Contents

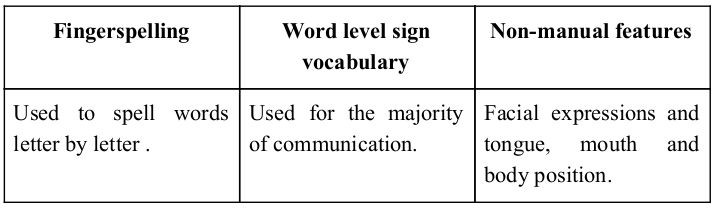
# **Abstract**

Sign language is one of the oldest and most natural form of language for communication, but there are lots of people who don’t know sign language and it is very difficult to find interpreters. So, we have come up with a solution to this problem. We have made a real time machine learning model to address this problem. In our model, user can make gestures and his live video would be given to the model as input and that model will recognize the gestures made and convert those gestures into text. These texts can also be understood by people who don’t know sign language.

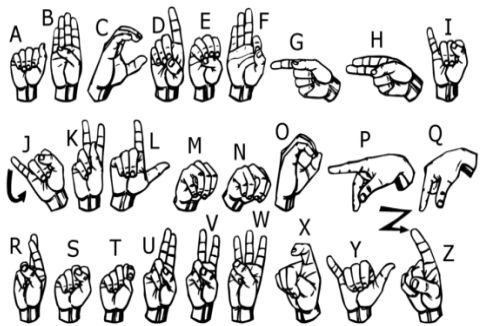
# **Introduction**

American sign language is one of the most famous sign languages around the globe. Communication is a major problem in the life of deaf and mute people and the only way they can communicate with others is the sign language. Communication is the process where 2 persons or multiple persons can exchange their thoughts and ideas in various ways such as speech signals, behavior and visuals. Deaf and mute people make use of their hands and expressions to convey what they want to speak and gestures are nonverbally exchanged messages and these can be understood with vision. This nonverbal communication of deaf and mute people is called sign language.

Sign language is a visual language and consists of 3 major components:



In our model we basically focus on producing a model which can recognize fingerspelling-based hand gestures. These gestures can combine to form a word, just like alphabets combine to form a word and those words can combine to form a sentence. The gestures we aim to train are given below:



# **Motivation**

A Communication barrier is developed between Deaf and Mute people and normal people as sign language is very much different from normal text. And people find it difficult to understand the sign language. So they depend on vision based communication for interaction.

If it is possible that there is a platform or application that can convert gestures into text then it will be very easy for other people to understand what people with speaking disability wants to convey. And that is where our project comes in mind.

The goal is to develop a user friendly and interactive interface that can understood the gestures made in various sign languages like American Sign Language (ASL), Indian Sign Language (ISL) and different other sign languages and convert those gestures into text so that these text can be easily understood by those who don’t know sign languages and there is no such communication barrier between normal people and the people who can’t speak or listen.

**Literature survey**

In the recent years there has been tremendous research done on the hand gesture recognition.

With the help of literature survey done we realized the basic steps in hand gesture recognition are :-

##### Data acquisition

##### Data preprocessing

##### Feature extraction

##### Gesture classification

### **Data acquisition:**

The different approaches to acquire data about the hand gesture can be done in the following ways:

### **Use of sensory devices**

It uses electromechanical devices to provide exact hand configuration, and position. Different glove based approaches can be used to extract information .But it is expensive and not user friendly.

### **Vision based approach**

In vision based methods computer camera is the input device for observing the information of hands or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describing

artificial vision systems that are implemented in software and/or hardware.

The main challenge of vision-based hand detection is to cope with the large variability of human hand’s appearance due to a huge number of hand movements, to different skin-colour possibilities as well as to the variations in view points, scales, and speed of the camera capturing the scene.

### **Data preprocessing and Feature extraction for vision-based approach:**

* We did not find any suitable dataset so we decided to make our own dataset by using or camera so as to train and test or machine learning model.
* We take photograph for each alphabet gesture in American Sign Language. For each alphabet we took around 65 images and that we did it for all the 26 alphabets.
* Images for all the 26 alphabets were stored in a folder and that folder consists of 26 different folders, one for each alphabet.
* We then used **Mediapipe** model to detect all the landmarks from the hand gestures pictures and store them as .np arrays.
* These are as would be used in the learning and training and testing of the models that will be making.

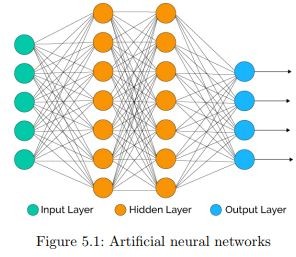
**Key Words and Definitions**

### **Feature Extraction and Representation** :

The representation of an image as a 3D matrix having dimension as of height and width of the image and the value of each pixel as depth (1 in case of Grayscale and 3 in case of RGB ). Further, these pixel values are used for extracting useful features using CNN.

### **Artificial Neural Networks :**

Artificial Neural Network is a connections of neurons, replicating the structure of human brain. Each connection of neuron transfers information to another neuron. Inputs are fed into first layer of neurons which processes it and transfers to another layer of neurons called as hidden layers. After processing of information through multiple layers of hidden layers, information is passed to final output layer.

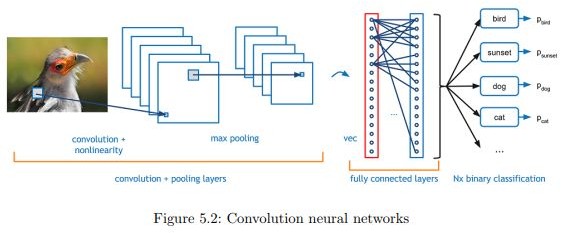


There are capable of learning and they have to be trained. There are different learning strategies :

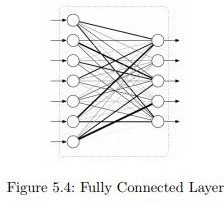
1. Unsupervised Learning
2. Supervised Learning
3. Reinforcement Learning

### **Convolution Neural Network :**

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.



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 color
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 :
   1. **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. Well lid this window and continue this process, so well finally get a activation matrix half of its original Size.
   2. **Average Pooling :** In average pooling we take 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, well connect the all the inputs to neurons.



1. **Final Output Layer :** After getting values from fully connected layer, well connect them to 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.

### **TensorFlow :**

Tensorflow is an open source software library for numerical computation. First we define the nodes of the computation actual computation takes place. TensorFlow is widely used in Machine Learning.

### **Keras** :

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.

### **OpenCV** :

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.

**Methodology**

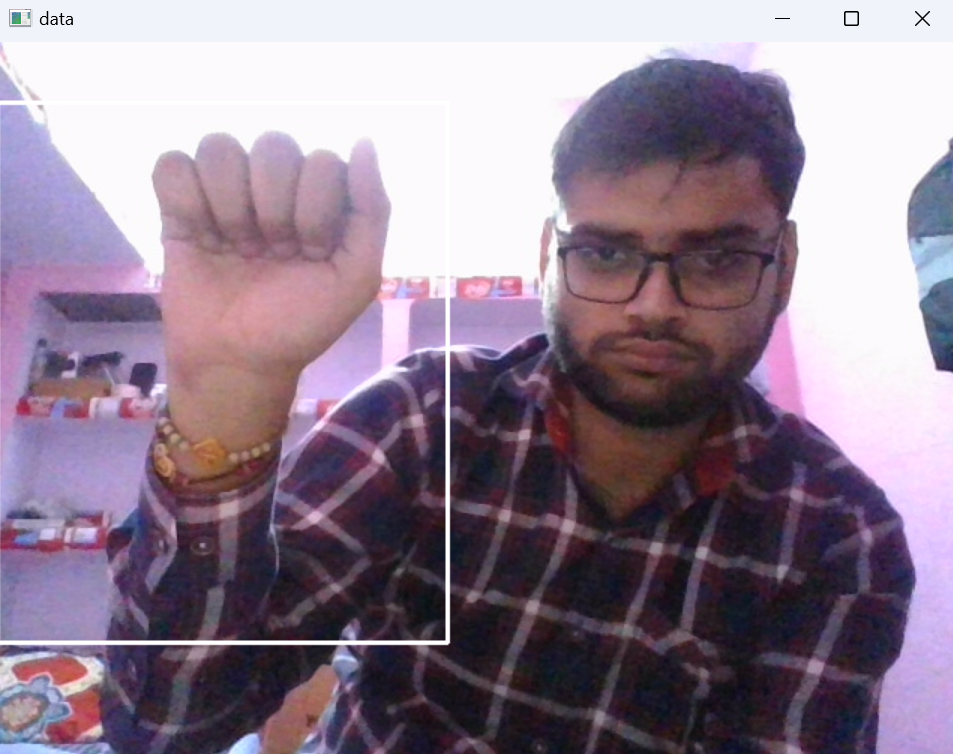
The system is a vision based approach. All the signs are represented with bare hands and so it eliminates the problem of using any artificial devices for interaction.

### **Data Set Generation**

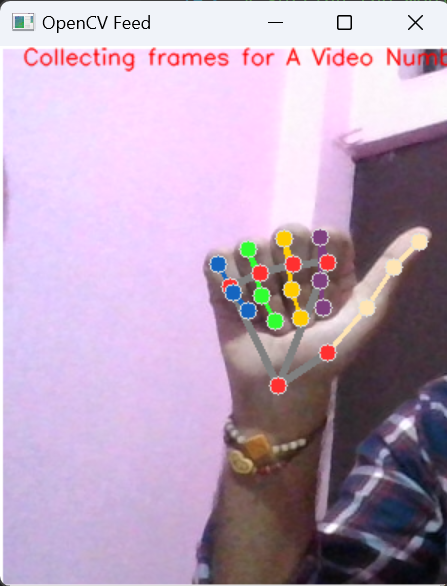
For the project we tried to find already made datasets but we couldn’t find dataset in the form of raw images that matched our requirements. All we could find were the datasets in the form of RGB values. Hence we decided to create our own data set. Steps we followed to create our data set are as follows.

We used Open computer vision(OpenCV) library in order to produce our dataset.Firstly we captured around 65 images of each of the symbol in ASL for training purposes and around 20 images per symbol for testing purpose.

First we capture each frame shown by the webcam of our machine. In the each frame we define a region of interest (ROI) which is denoted by a white bounded square as shown in the image below.



From this whole image we extract our ROI which is landmarks of our hand while making the gestures and store them as numerical values in Numpy arrays.



### **GESTURE CLASSIFICATION**

###### The approach which we used for this project is :

###### The .np arrays containing the landmarks for hands are transferred to the LSTM model for prediction and is a letter is detected for more than 20 frames then the letter is printed.

###### Also 3 dense layers are also applied to enhance the output received from the upper 3 LSTM layers

# **Challenges Faced** :

There were many challenges faced by us during the project. The very first issue we faced was of dataset. We wanted to deal with raw images and that too square images as CNN in Keras as it was a lot more convenient working with only square images. We couldn’t find any existing dataset for that hence we decided to make our own dataset. Second issue was to select a model which we could apply on our images so that proper features of the images could be obtained and hence then we could provided that image as input for CNN model. We tried various models but finally we settled with **Mediapipe**. More issues were faced relating to the accuracy of the model we trained in earlier phases which we eventually improved by increasing the input image size and also by improving the dataset.

# **Results :**

We have achieved an accuracy of 65% in our model using which is a better accuracy for beginners. Most of the research papers focus on using devices like kinect for hand detection. Some researchers build a recognition system for flemish sign language using convolutional neural networks and kinect and achieve an error rate of 2.5%. Another researcher made a recognition model using hidden markov model classifier and a vocabulary of 30 words and they achieve an error rate of 10.90%. In another report, they achieve an average accuracy of 86% for 41 static gestures in japanese sign language. Using depth sensors map, people achieved an accuracy of 99.99% for observed signers and 83.58% and 85.49% for new signers. 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..

# **Conclusion :**

In this report, a functional real time vision based american sign language recognition for D&M people have been developed for asl alphabets.We achieved final accuracy of 65.0% on our dataset. We could improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other.

This way we would be able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.

.

# **Future Scope :**

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 preprocessing to predict gestures in low light conditions with a higher accuracy.

**References :**

[1]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.

* 1. Pujan Ziaie, Thomas M ̈uller , Mary Ellen Foster , and Alois Knoll“A Na ̈ıve Bayes Munich,Dept. of Informatics VI, Robotics and Embedded Systems,Boltzmannstr. 3, DE-85748 Garching, Germany.
  2. https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian\_median\_blur\_b ilateral\_filter/gausian\_median\_blur\_bilateral\_filter.html
  3. Mohammed Waleed Kalous, Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.
  4. aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convol utional-Neural-Networks-Part-2/
  5. <http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php>
  6. Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925.

Springer, Cham

* 1. Zaki, M.M., Shaheen, S.I.: Sign language recognition using a combination of new vision based features. Pattern Recognition Letters 32(4), 572–577 (2011)
  2. N. Mukai, N. Harada and Y. Chang, "Japanese Fingerspelling Recognition Based on Classification Tree and Machine Learning," *2017 Nicograph International (NicoInt)*, Kyoto, Japan, 2017, pp. 19-24. doi:10.1109/NICOInt.2017.9
  3. Byeongkeun Kang , Subarna Tripathi , Truong Q. Nguyen ”Real-time sign language fingerspelling recognition using convolutional neural networks from depth map” 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)
  4. <https://opencv.org/>
  5. <https://en.wikipedia.org/wiki/TensorFlow> [13]<https://en.wikipedia.org/wiki/Convolutional_neural_network>

# **APPENDIX**

### **OpenCV**

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

### **Convolution Neural network**

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

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 the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

**Tensorflow**

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. 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 2.0 open source library on November 9, 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.