Sign Language to Text Conversion for the Dumb and Deaf

EE 272 Exploratory Project

Under the supervision of - Ms. Shobhita Meher ,Assistant Professor,Department of Electrical Engineering, IIT(BHU)

By:
Ayush Raj - 19085018
Karan Aditya Singh Bishnoi - 19085046
Aanchal Jain - 19085002

Abstract

Sign language is one of the oldest and most natural forms of language for communication, but since most people do not know sign language and interpreters are very difficult to come by we have come up with a real time American sign language detection method using computer vision. In our method, the image of the hand is passed through a classifier which predicts the class of the hand gestures. Our method provides 99.86 % accuracy for detection of the 24 letters of the alphabet (J and Z are excluded as they can be detected only through videos and not images) with signs for ‘delete’ and ‘space’ as well.

Introduction

American Sign Language (ASL) is a complete, complex language that employs signs made with the hands and other movements, including facial expressions and postures of the body. It is the first language of many deaf North Americans, and one of several communication options available to deaf people. ASL is said to be the fourth most commonly used language in the United States. It is a predominant sign language since the only disability deaf and dumb 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, behavior and visuals. Deaf and dumb people make use of their hands to express different gestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language. Sign language is a visual language and consists of 3 major components :

Fingerspelling	**Word level sign vocabulary**	**Non-manual features**
Used to spell words letter by letter	Used for the majority of communication	Facial expressions and tongue, mouth and body position.

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 detect are as given in the image below.

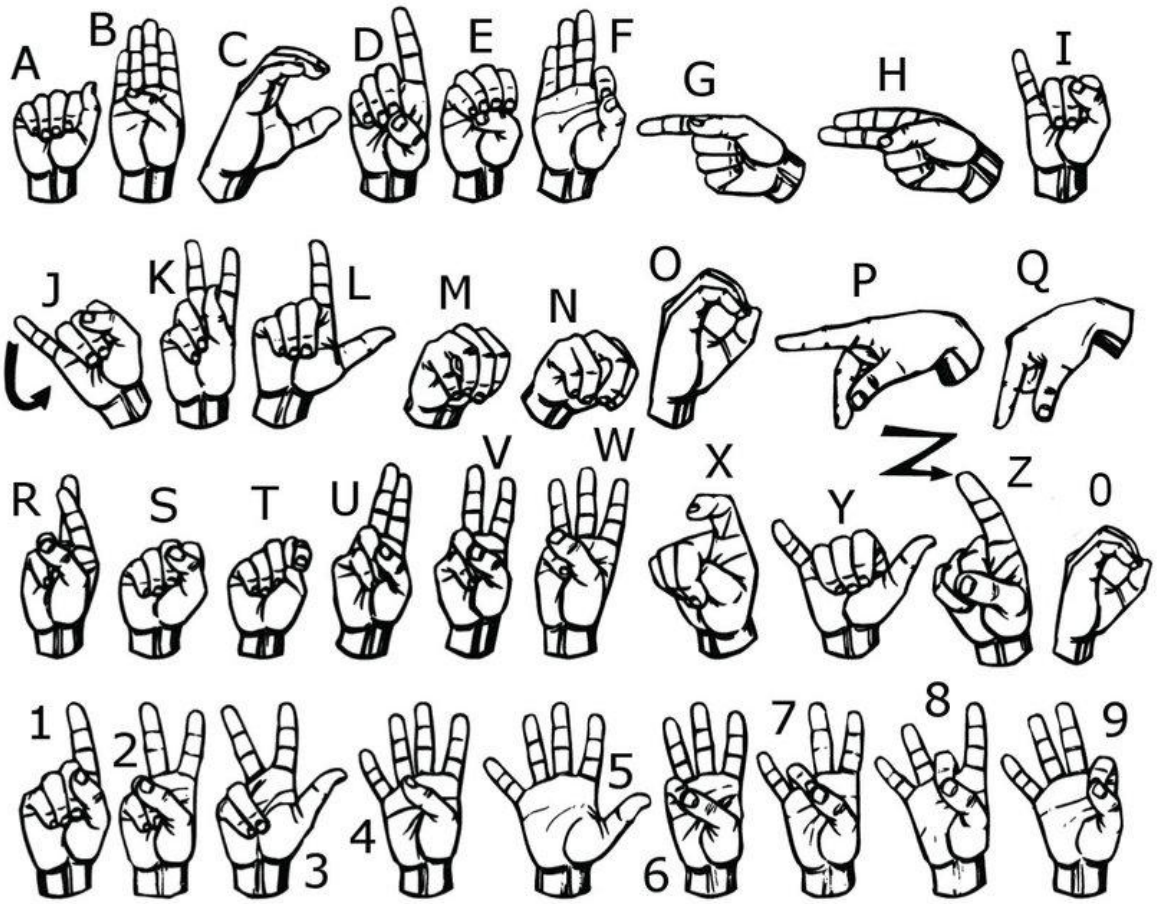

Fig 1 : ASL signs

Motivation

For interaction between normal people and **Deaf and dumb** 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 such people can enjoy communication without really knowing each other's language.

The aim is to develop a user-friendly human computer interface 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. We chose American Sign Language as it is so far most widely used.

Literature Survey

In recent years there has been tremendous research done on hand gesture recognition. With the help of literature survey done we have listed the basic steps in hand gesture recognition -

- Data acquisition
- Data preprocessing
- Gesture classification

• Data acquisition:

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

1. 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.

2. Vision based approach

In vision based methods, a 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. 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:

It includes transforming the image data obtained in such a way that they are compatible with the further methods and also contain useful information, leaving behind unnecessary information (called noise). In case of image processing for hand gesture detection, it includes converting images to matrices, image resizing and removing noise from the images.

- Gesture classification :

Several ways of gesture classification is listed as follows -

1. Hidden Markov Models (HMM) can be used for the classification of the gestures. This model deals with dynamic aspects of gestures.Gestures are extracted from a sequence of video images by tracking the skin-colour blobs corresponding to the hand into a body– face space centered on the face of the user. The goal is to recognize two classes of gestures: deictic and symbolic.The image is filtered using a fast look–up indexing table. After filtering, skin colour pixels are gathered into blobs. Blobs are statistical objects based on the location (x,y) and the colourimetry (Y,U,V) of the skin colour pixels in order to determine homogeneous areas.

2. Naive Bayes Classifier can also be used which is an effective and fast method for static hand gesture recognition. It is based on classifying the different gestures according to geometric based invariants which are obtained from image data after segmentation.Thus,unlike many other recognition methods, this method is not dependent on skin colour. The gestures are extracted from each frame of the video,with a static background. The first step is to segment and label the objects of interest and to extract geometric invariants from them. Next step is the classification of gestures by using a K nearest neighbor algorithm aided with distance weighting algorithm to provide suitable data for a locally weighted Naive Bayes classifier.

3. We can also construct a skin model to extract the hand out of an image and then apply image processing techniques to remove noise from the image. After obtaining the noise free image, we can use these images to train a convolutional neural network model in order to predict the outputs. Using convolutional neural networks or models based on convolutional neural networks has always shown good results in computer vision applications. Interfaces such as Tensorflow, Keras and Pytorch have made their implementation easier.

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.

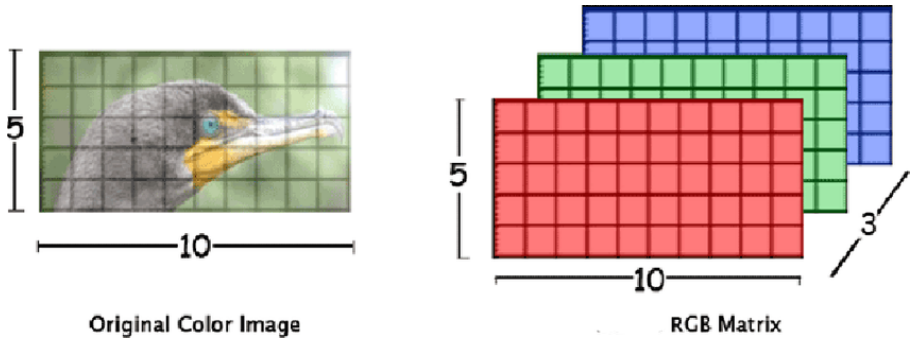

Fig 2 : Representation of a colour image into RGB Matrix

- Artificial Neural Networks :

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

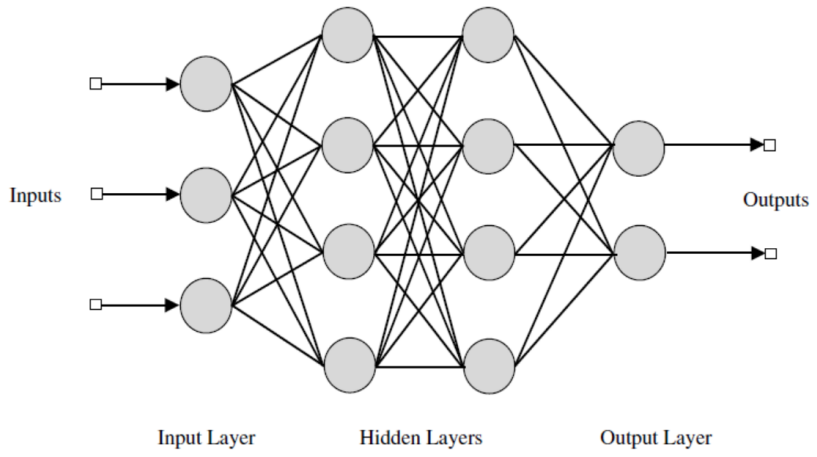

Fig 3 : an example of artificial neural network

They 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 (CNN) :

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.

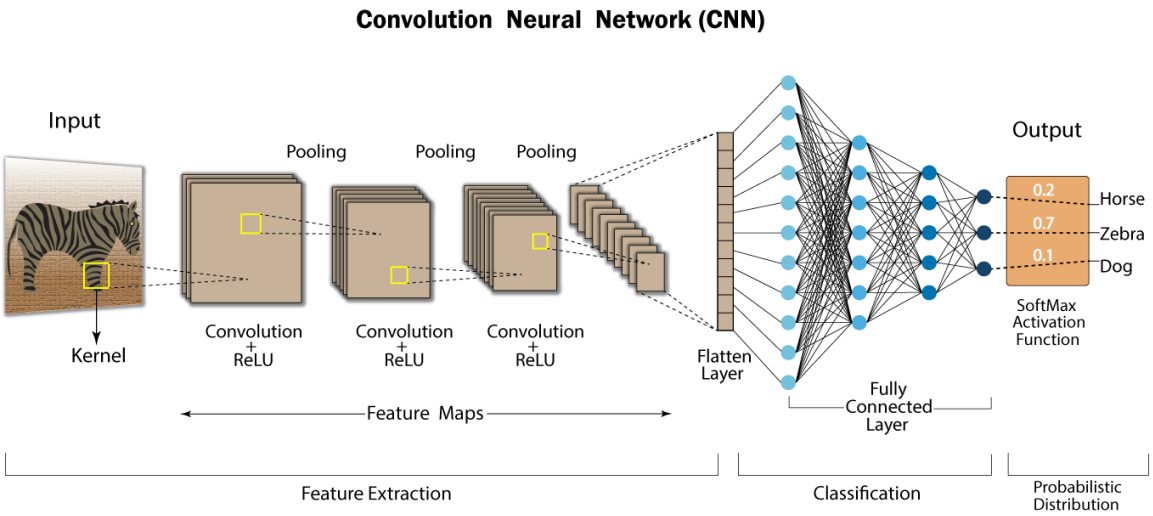

Fig 4 : an example of convolutional neural network

1. Convolution Layer : In convolution layer we take a small window size (typically of length 3*3 or 5*5) that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration we slide the window by stride size (typically 1 or 2), and compute the dot product of filter entries and input values at a given position. As we continue this process we will 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 a pooling layer to decrease the size of the 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 will slide this window and continue this process, so we will 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.

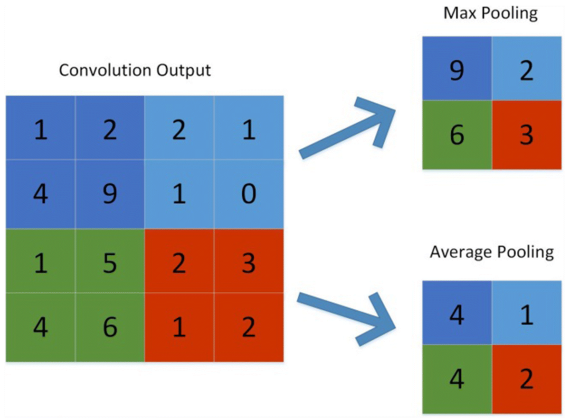

Fig 5 : Representation of the working of pooling

3. Fully Connected Layer : In convolution layer neurons are connected only to a local region, while in a fully connected region, we will connect all the inputs to neurons. They are also called densely connected layers.

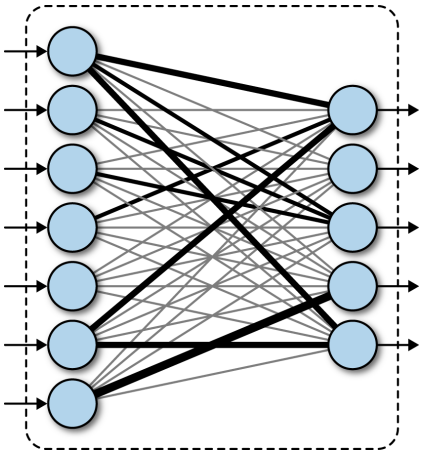

Fig 6 : Representation of the fully connected layers

4. Activation function : the activation function of a node defines the output of that node given an input or set of inputs. Some examples of activation function generally used in Convolution neural networks are -

a. **Rectified Linear Unit (ReLU) :** The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

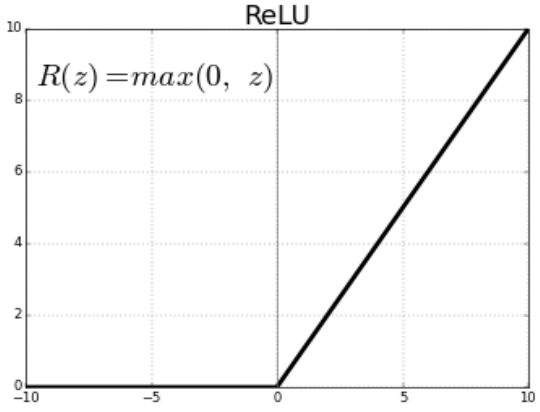

Fig 7 : Representation of ReLU activation function

b. **Softmax :** Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

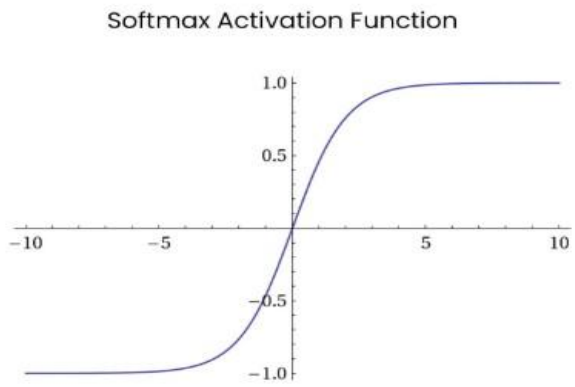

Fig 8 : Representation of Softmax activation function

5. Final Output Layer : After getting values from the fully connected layer, we will 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.

- 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. 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 easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

- 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

1. Data Set Generation

We used the vision based approach to generate the dataset. So the sole requirement for dataset generation was a camera. A general mobile or laptop camera was good enough. For the project we tried to find already made datasets but we couldn’t find a dataset in the form of raw images that matched our requirements. We could find the datasets in the form of RGB values. Some image datasets were available but classifier models trained on them didn't give satisfactory

results on testing. Hence we decided to create our own data set.

We used the open computer vision (OpenCV) library in order to produce our dataset. We captured around 1000 - 1500 images of each of the symbols in ASL. We used around 80% of the dataset for training and rest 20% for testing. We captured each frame shown by the webcam of our machine or mobile camera. While capturing images from the webcam ,in each frame we defined a region of interest (ROI) and showed the hand inside the ROI. Below are some images of the dataset created -

Fig 9 : A set of images taken for the dataset

As these images were of different size, all the images were reshaped to (64, 64, 3) for training. Further these images were converted into grayscale images as colour doesn't play any importance in hand gesture classification. So the final image shape used for training was (64, 64, 1). A set of final resized and grey-transformed images is shown below -

Fig 10 : A set of final images of all letters in the dataset

2. Gesture Classification

We used the following algorithm to classify and print sign language labels -

1. Apply grayscale filter to the frame taken and resize it 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 the majority of times based on a threshold value then the letter is printed and taken into consideration for forming the word, i.e. added to the resultant string.

3. If the predicted label is ‘Space’, a space is added to the resultant string. If the predicted label is ‘Del’ and the resultant string contains at least one character, the last letter of the resultant string is removed.

CNN Model :

The first layer of the CNN model is a convolution 2D layer of 16 filters. It takes the image of size (64, 64, 1). It is followed by convolution 2D layers of number of filters 32 and 64 respectively. Then follows a Maxpooling2D layer of pool size (3, 3). After this, three convolution 2D layers of filters 64, 128, 254 respectively are added, again followed by a Maxpooling2D layer of pool size (3, 3). Then finally the last triplet of convolution 2D layers of filters 254, 512 and 1024 respectively are added. This is again followed by the Maxpooling2D layer of pool size (3, 3). After this, a batch normalisation layer is added.

The shape of the image after the batch normalisation layer is (2, 2, 1024). After this the image is flattened in order to feed it into dense layers. A dropout layer is added of dropout value 0.2 . After this a dense layer of size 1024 is

added. Finally the output layer (dense layer) is added of size 27 (i.e. total number of classes for classification). The visual representation of the model is given below -

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 16)        160
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 21, 21, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 64)        36928
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 21, 21, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 21, 21, 256)       295168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 256)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 512)         1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 7, 7, 1024)        4719616
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 1024)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 2, 2, 1024)        4096
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              4195328
_________________________________________________________________
dense_1 (Dense)              (None, 27)                27675
=================================================================
Total params: 11,146,203
Trainable params: 11,144,155
Non-trainable params: 2,048
```

Fig 11 : Model Architecture

Activation Function :

We have used ReLu (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected layers) except the output layer. This adds nonlinearity to the formula and helps to learn more complicated features.It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time. The activation function used in the output layer is softmax.

Pooling Layer :

We apply Max pooling to the input image with a pool size of (3,3). This reduces the amount of parameters thus lessening the computation cost and reduces overfitting.

Dropout Layers:

The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples.This layer "drops out" a random set of activations in that layer by setting them to zero. The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out. We used the dropout layer once with dropout value 0.2.

Optimizer :

We have used Adam optimizer for updating the model in response to the output of the loss function. Adam combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp). We used 'categorical_crossentropy' as the parameter to define loss.

3. Finger spelling sentence formation

Implementation :

We maintain a dictionary with letters as key and their respective frequencies as the value. After taking a frame from the webcam and getting a predicted letter, we add unity to the frequency of that letter. If the predicted letter is ‘Nothing’, we again initialise the dictionary with all values as zero and continue with the next frame from the webcam. If the sum of frequencies of all the letters is less than 60, we don’t add anything to the resultant string. This acts as a threshold check for the letter prediction. After the sum of frequencies of all the letters exceeds 60, we keep checking if a letter attains majority frequency. We define a letter to have attained majority if its frequency is more than or equal to half of the sum of frequencies of all the letters. If a letter attains majority, we add it to the resultant string and initialise the dictionary with all values as zero. If the predicted majority letter is ‘Del’ and there is at least one letter in the resultant string, we remove the last letter of the resultant string.

Training and Testing :

We convert our input images (RGB) into grayscale and resize our images to 64*64. We feed the input images after preprocessing 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 value in each class sums to 1. We have achieved this using the softmax activation function. We divided the created dataset into training, validation and testing dataset in the ratio of 3 : 1 : 1. We used the validation dataset to come up with the best set of hyperparameters used to train in order to attain best validation accuracy. We then tested the model with the test dataset.

Results

We have achieved test accuracy of 99.86% in our model. The model is able to classify the majority of the letters well. The accuracy vs epochs and loss vs epochs graph is shown below -

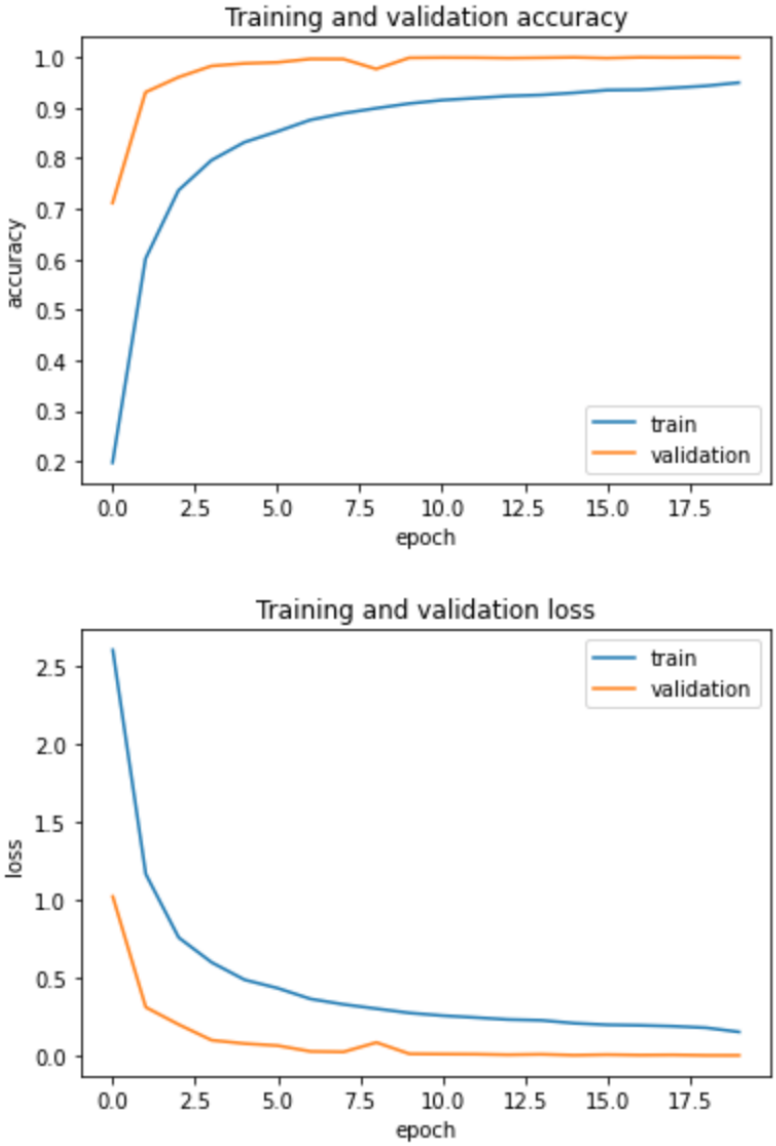

Fig 12 : Loss and accuracy graph

Challenges Faced

There were many challenges faced by us during the project. The very first issue we faced was the dataset. We couldn't find any existing dataset that gave satisfactory results for that hence we decided to make our own dataset. More issues were faced relating to the accuracy of the model we trained in earlier phases which we eventually improved by decreasing the input image size and other hyperparameter tuning and also by improving the dataset. Doing the project in the online semester is another challenge as communication and reachability with team members has been reduced.

Conclusion

In this report, a functional real time vision based American Sign Language recognition for the deaf and dumb people have been developed for english alphabets. We achieved final accuracy of 99.86% on our dataset.

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

Future Scope

The project can be further improved to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. The scope of the project can be increased by improving the preprocessing to predict gestures in low light conditions with a higher accuracy. The project can be further improved by adding some object localization algorithm like YOLO to localize hand gestures instead of using an already defined ROI.

References

1. https://www.tensorflow.org/tutorials/images/classification
2. https://keras.io/guides/
3. https://www.kaggle.com/grassknoted/asl-alphabet
4. https://en.wikipedia.org/wiki/American_Sign_Language
5. https://opencv-python-tutroals.readthedocs.io/en/latest/
6. https://jupyter-notebook.readthedocs.io/en/stable/
7. https://colab.research.google.com/
8. https://pypi.org/project/PySimpleGUI/