A Deep Learning Mobile Application based Sign Language Recognition for Aphasic Person

Hrishikesh Jadhav[1], Pushkar Dounde[2], Akash Pawar[3], Abhishek Muthange[4]

Department of Computer Engineering, Sinhgad College of Engineeing, Vadgaon, Pune[1234]

ABSTRACT

Pervasive sign language is used relatively within our society, deaf and other verbally challenged communication of people with hearing impairments is of prime importance in our society. This paper presents a solution for this purpose. We developed an android application which integrates both computer vision techniques involving Histogram of Oriented Gradients (HOG) as well as Machine Learning techniques such as Convolutional Neural Network (CNN) and Multiclass Support Vector Machine (SVM) to detect and recognize gestures automatically. People tend to face difficulty in communicating on daily basis. Because of that development of assistive, cost-effective, as well as non-invasive technologies to aid the people in need is necessity.

KEYWORDS

Sign Language, Hand Gesture Recognition, Image Processing, Grey Scale image, Convolution Neural Network

I. INTRODUCTION

Development in computer technology is increasing, as the need for accord amongst people and machines. Communication is one of the most important aspects of the human development, and increment in technological development is helping to improve that aspect. Effective communication is a very important skill which helps us to understand and connect with the people around us. It allows us to build respect and trust, resolve differences and maintain sustainable development in our environment where problem solving, caring and creative ideas can thrive. For any argument or conflict in any relationship, most of the times poor communication skills are responsible. Inattentiveness, arguments, vilification and language barrier between the communicators are the primary indicators of poor communication. All of these factors not only affect the physically fit people but also affect the physically challenged. And investigating the barrier of communication between the hearing-impaired and the hearing person has led to the need of providing a means of bridging this communication gap.

Sign language is a form of hand gestures involving visual motions and signs, which is used as a system of communication by hearing-impaired as well as verbally challenged people. However there are not that many people who are able to understand sign languages, and this poses a genuine communication barrier between the deaf community and the rest of the society.

This paper sheds light on the recognition of letters from the hand gestures which is taken as per the American Sign Language. The detection is done using various techniques of Image Processing and Feature extraction. The paper invokes various computer vision techniques as well as numerous algorithms which are used in gesture recognition and further processing. Python prototype is used for image processing and deep learning, and it is integrated into an Android application for remote functionality. The project basically focuses on producing a model which can recognise Finger spelling 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.

Fig. 1: American Sign Language Alphabets

II. MOTIVATION

A language bridge is created between normal people and aphasic people in the form of a sign language structure that differs from normal text in order to facilitate contact. As a result, for contact, they depend on vision-based communication. The signs can be readily interpreted by other people if there is a common GUI that translates the sign language to text. As a result, research has been conducted on a vision-based GUI device that allows Aphasic people to communicate without having to know each other's language. The aim is to create a user-friendly human-computer interface (HCI) in which the device recognizes human sign language. There are many different sign languages used throughout the world, including American Sign Language (ASL). There are many sign languages used throughout the world, including American Sign Language (ASL), French Sign Language (FSL), British Sign Language (BSL), Indian Sign Language, and Japanese Sign Language, among others.

III. LITERATURE SURVEY

Hand gesture recognition has gotten a lot of attention in recent years.

We discovered the basic steps in hand gesture recognition using the results of a literature review :-

- Data acquisition
- Data pre-processing
- Feature extraction
- Gesture classification

A. *Data Acquisition*

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

1. Use of sensory devices

Electromechanical instruments are used to provide precise hand tuning and positioning. To retrieve information, various glove-based methods can be used. However, it is both costly and inconvenient to use.

2. Vision based approach

A computer camera is used as an input system in vision-based approaches to observe detail from hands or fingers. The Vision Based approaches only involve a camera, allowing for normal human-computer interaction without the use of any additional equipment. By defining artificial vision systems that are applied in software and/or hardware, these systems aim to supplement biological vision.

The biggest task in vision-based hand recognition is to deal with the wide range of human hand appearances caused by a vast variety of hand gestures, various skin colour options, and changes in viewpoints, scales, and pace of the camera recording the scene.

B. *Data preprocessing and Feature extraction for vision based approach*

- The method for detecting hands in [1] incorporates threshold-based colour recognition and context subtraction. Faces and hands have identical skin colours, but we can use Adaboost face detector to distinguish between them. Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- We may also use a filter called Gaussian blur to remove the appropriate picture for training. The filter is defined in [3] and can be easily implemented using open computer vision, also known as OpenCV.
- We can use instrumented gloves to extract the appropriate image that will be taught, as stated in [4]. As opposed to adding filters to data received from video extraction, this helps to minimize the computing time for pre-processing and can give us more succinct and reliable data.
- We tried using colour segmentation techniques to do hand segmentation of an image, but as mentioned in the research paper, skin colour and tone are highly dependent on lighting conditions, so the results we got for the segmentation we attempted were not very good. Furthermore, we have a large number of symbols to train for our project, many of which are identical in appearance, such as the gesture for the symbol "V" and the digit "2," so we agreed that in order to achieve better accuracies for our large number of symbols, we would use a different method, rather than segmenting the hand from a random backdrop, we keep the hand's background a consistent single colour so we don't have to section it by skin colour. This will assist us in achieving greater results.

C. *Gesture classification*

- The description of movements is done using Hidden Markov Models (HMM) in [1. The dynamic dimensions of gestures are addressed in this model. The skin-colour blobs referring to the hand are tracked into a body–face space based on the user's face to remove gestures from a series of video images. The aim is to distinguish between two types of movements: deictic and symbolic gestures. A quick look–up indexing table is used to filter the image. Skin colour pixels are collected into blobs during filtering. Blobs are mathematical artifacts used to determine homogeneous areas depending on the position (x, y) and colourimetry (Y,U,V) of skin colour pixels.
- The Nave Bayes Classifier is used in [2], which is an accurate and quick tool for recognizing static hand gestures. It works by categorizing various movements using geometric invariants extracted from image data after segmentation. As a result, unlike many other systems of identification, this one is not based on skin colour. For a static context, the movements are derived from each frame of the film. The first step is to slice, mark, and remove geometric invariants from the properties of interest. To have appropriate data for a locally weighted Nave Bayes classifier, the next step is to classify movements using a K nearest neighbour algorithm assisted with distance weighting algorithm (KNNDW).
- According to paper on "Human Hand Gesture Recognition Using a Convolution Neural Network" by Hsien-I Lin , Ming-Hsiang Hsu, and Wei-Kai Chen graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan, they construct a skin model to extract the hand out of an image and then apply binary threshold to the whole image. After obtaining the threshold image they calibrate it about the principal axis in order to center the image about it. They input this image to a Convolutional Neural Network model in order to train and predict the outputs. They have trained their model over 7 hand gestures and using their model they produce an accuracy of around 95% for those 7 gestures.

IV. DEFINITIONS

A. *Feature Extraction and Representation*

The representation of an image as a three-dimensional matrix of height and width dimensions and depth values for each pixel (1 in case of Grayscale and 3 in case of RGB). These pixel values are also used by CNN to retrieve valuable features

B. *Artificial Neural Networks*

Artificial Neural Network is a connection 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.

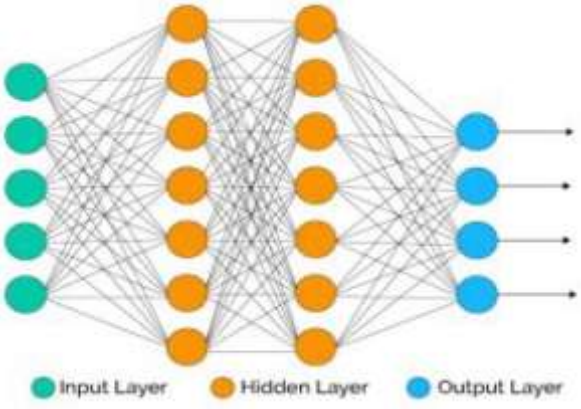

Fig. 2: Artificial Neural Network Architecture

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

C. Convolution Neural Network

In contrast to normal Neural Networks, the neurons in CNN layers are organized in three dimensions: distance, height, and depth. Instead of connecting all the neurons in a layer in a totally connected way, the neurons in a layer would only be connected to a specific region of the layer (window size) before it. Furthermore, since we can minimize the entire picture into a single vector of class scores by the end of the CNN architecture, the final output layer will have dimensions (number of classes).

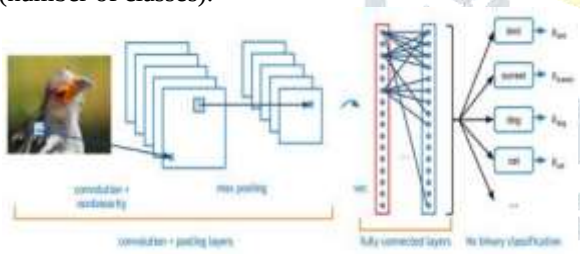

Fig. 3: Convolutional Neural Network Architecture

1) Convolution Layer: We use a limited window size [typically 5*5] in the convolution layer that stretches to the depth of the input matrix. The layer is made up of learnable window size filters. We slid the window by stride size [typically 1] at each iteration, and computed the dot product of filter entries and input values at a given location. Build a 2-Dimensional activation matrix that gives the answer of that matrix at any spatial location as we begin this procedure. That is, the network can learn filters that function when certain visual features, such as an edge of a certain orientation or a blotch of a certain colour, are detected.

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

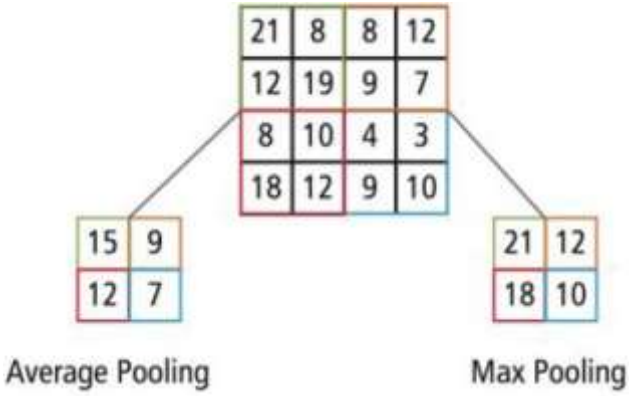

Fig. 4: Pooling Layer in CNN

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.

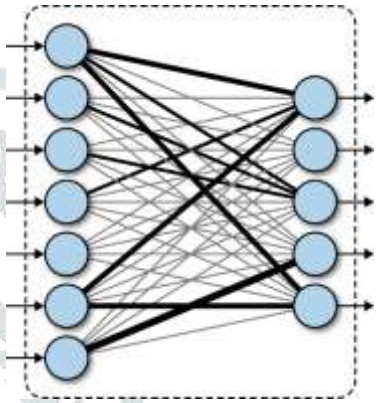

Fig. 5: Fully Connected Deep Networks layer

4) 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.

D. TensorFlow

TensorFlow is an open-source software library for numerical computation. First, we define the nodes of the computation graph, and then inside a session, the actual computation takes place. TensorFlow is widely used in Machine Learning.

E. Keras

Keras is a Python library for high-level neural networks that acts as a wrapper for TensorFlow. It is useful when we need to create and validate a neural network easily and with few lines of code. It includes implementations of typical neural network elements such as layers, objectives, activation functions, optimizers, and tools for dealing with images and text data.

F. 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, and MATLAB/OCTAVE.

V. METHODOLOGY

The system is based on the concept of vision. All of the signs are made using one's own fingertips, which prevents the need for any electronic equipment for touch.

A. Dataset Generation

We looked for pre-made datasets for the project but could not find any that met our specifications in the form of raw images. The datasets in the form of RGB values were the only ones we could find. As a result, we wanted to make our own data collection. The following are the steps we took to build our data collection.

To build our dataset, we used the Open Computer Vision (OpenCV) library. To begin, we took approximately 800 images of each ASL symbol for training purposes and approximately 200 images for testing purposes.

1) We begin by capturing each frame shown by our machine's webcam. We describe a region of interest (ROI) in each frame, which is represented by a blue bounded square in the image below.

Fig. 6: RGB image with Region of Interest

2) We strip our ROI, which is RGB, from the whole image and transform it to grayscale as seen below.

Fig. 7: Grey Scale Image

3) Finally, we add our gaussian blur filter to our image, which aids in the extraction of various features. Since adding gaussian blur, the picture looks like this.

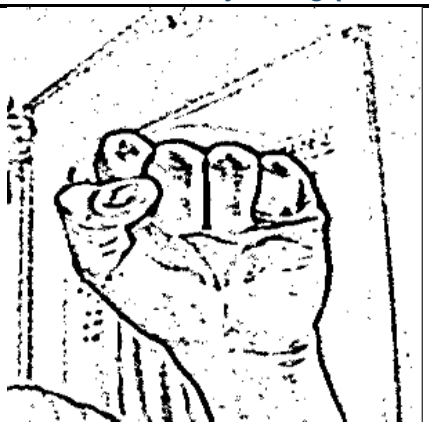

Fig. 8: Gaussian Blur Filter

B. Gesture Classification

Our approach uses one layers of algorithm to predict the final symbol of the user.

1) CNN Model:

a) *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.

b) *1st Pooling Layer:* The pictures are downsampled using max pooling of 2x2 i.e we keep the highest value in the 2x2 square of array. Therefore, our picture is downsampled to 63x63 pixels.

c) *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.

d) *2nd Pooling Layer :* The resulting images are downsampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.

e) *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 30x30x32 =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.

f) *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.

g) *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).

2) Activation Function*:*

We have used ReLu (Rectified Linear Unit) in each of the layers (Convolutional as well as fully connected neurons). ReLu calculates max(x, 0) for each input pixel. 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.

3) Pooling Layer:

We apply Max pooling to the input image with a pool size of (2, 2) with relu activation function. This reduces the

amount of parameters thus lessening the computation cost and reduces overfitting.

4) 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 does not 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 [5].

5) 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).

C. Traning and Testing

To minimize unwanted noise, we transform our RGB input images to grayscale and use Gaussian blur. To remove our hand from the frame, we use adaptive threshold and scale our images to 128 x 128 pixels.

We feed the preprocessed input images to our model for training and testing after performing all of the above operations.

The prediction layer calculates the likelihood of the picture falling into one of the groups. As a result, the output is scaled between 0 and 1, and the sum of each class's values equals 1. Using the SoftMax function, we were able to accomplish this.

The prediction layer's contribution will initially be a little off from the actual value. To improve it, we used labeled data to train the networks. Cross-entropy is an efficiency metric that is used in classification. It is a continuous equation that is positive when the value is not the same as the labeled value and zero when the value is the same as the labeled value. As a result, we improved the cross-entropy by bringing it as close to zero as possible. We change the weights of our neural networks in our network layer to accomplish this. The cross entropy can be calculated using TensorFlow's built-in function.

We used Gradient Descent to optimize the cross-entropy equation after discovering it. The best gradient descent optimizer is called Adam Optimizer.

CHALLENGES FACED

Throughout the project, we faced numerous challenges. The first problem we ran into was a lack of details. We preferred to work with raw images, specifically square images, since working with only square images was much more convenient in Keras. Since we couldn't find an available dataset for that, we wanted to build our own. The second problem was to choose a filter that we could add to our images in order to extract the images' proper characteristics, which we could then use as data for the CNN model. We experimented with a variety of filters, including binary threshold, canny edge detection, and Gaussian blur, but ultimately chose the Gaussian blur filter. More problems with the model's precision were encountered earlier in the process, which we ultimately resolved by increasing the input image size and optimizing the dataset.

RESULTS

We obtained 95.8% accuracy in our model using just layer 1 of our algorithm, which is higher than most existing study papers on American Sign Language. Most of the research papers focus on using devices like kinect for hand detection. The majority of the research papers concentrate on using kinect-like instruments to sense hands. In [7], they use Convolutional neural networks and kinect to build a recognition scheme for Dutch sign language with a 2.5 percent error rate. [8] Uses a secret markov model classifier and a vocabulary of 30 terms to construct a recognition model with a 10.90 percent error rate. In [9], they reach an overall precision of 86 percent in Japanese sign language for 41 static gestures. Chart [10] reached an accuracy of 99.99 percent for observed signers and 83.58 percent and 85.49 percent for new signers using depth sensors. Their identification scheme was also based on CNN. One factor to keep in mind is that our model does not use a context subtraction algorithm, although some of the other models do. As a result, the accuracy of history subtraction can vary depending on how we apply it in our project. However, although the majority of the above projects make use of Kinect instruments, our key goal were to build a project that could be implemented using readily available tools. Since a sensor like kinect is not only not widely available but also pricey for much of the audience, our model, which uses a standard laptop webcam, is a huge bonus. The confusion matrices for our results are shown below.

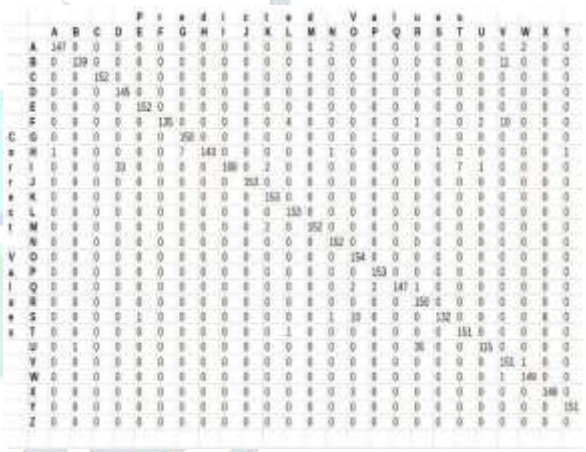

Fig. 9: Final Resulting Confusion Matrices

CONCLUSION

In this study, an American Sign Language alphabet-based practical real-time vision-based American Sign Language recognition system for aphasic people is created. On our dataset, we obtained a final accuracy of 95.0 percent. Since applying two layers of algorithms, we are able to enhance our prediction by verifying and predicting symbols that are more close to one another. We can distinguish nearly all symbols this way if they are displayed correctly, there is no background noise, and the illumination is sufficient.

FUTURE SCOPE

We plan to test various background subtraction algorithms in order to achieve higher accuracy, even in the case of complex backgrounds. We are also considering enhancing the preprocessing so that we can more accurately predict movements in low-light situations.

REFERENCES

[1] Mohammed Elmahgiubi, Mohamed Ennajar, Nabil Drawil, Mohamed Samir Elbuni, "Sign Language Translator and Gesture recognition", p. 2015

[2] Neha Poddar, Shrushti Rao, Shruti Sawant, Vrushali Somavanshi, Prof. Sumita Chandak, “Study of Sign Language Translation using Gesture Recognition”, p. 2015

[3] K. Manikandan, Ayush Patidar, Pallav Walia, Aneek Barman Roy,, “Hand Gesture Detection and Conversion to speech and text”, p. 2018

[4] 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.

[5] Pujan Ziaie, Thomas M¨uller , 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.

[6] https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_b ilateral_filter/gausian_median_blur_bilateral_filter.html

[7] https://aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convol utional-Neural-Networks-Part-2/

[8] http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php

[9] 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

[10] 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)

[11] 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

[12] 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)

[13] https://opencv.org/

[14] https://en.wikipedia.org/wiki/TensorFlow

[15] https://en.wikipedia.org/wiki/Convolutional_neural_network