IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Conversion of Sign Language to Text for Deaf and Dumb

[1]Dr B Esther Sunanda, [2]Ch Sravya, [3]Ch Satyavani, [4]Ch Geethika, [5]Ch Aruna Sri
[1] Dept. of Computer Science and Systems Engineering
[1]Andhra University, Visakhapatnam, India

Abstract : Sign Language is one of the oldest and most natural form of language for communication, but since most people do not know sign language and interpreters are very difficult to come by come by, we have come up with a real time method using neural networks for fingerspelling based American sign Language. In our method, the hand is first passed through a filter and after the filter is applied, the hand is passed through a classifier which predicts the class of the hand gestures. Our method provides 95.7% accuracy for the 26 letters of the alphabet.

IndexTerms **– Sign Language**

I.INTRODUCTION

American sign language is a predominant sign language Since the only disability D&M people have is communication related and they cannot usespoken languages hence the only way for them to communicate is throughsign language. Communication is the process of exchange of thoughts andmessages in various ways such as speech, signals, behaviour and visuals. Deaf and dumb(D&M) people make use of their hands to express differentgestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbally communication of deaf and dumb people is calledsign language.
Minimizing the verbal exchange gap among D&M and non-D&M people turns into a want to make certain effective conversation among all. Sign language translation is among one of the most growing lines of research and it enables themaximum natural manner of communication for those with hearing impairments.A hand gesture recognition system offers an opportunity for deaf people to talk with vocal humans without the need of an interpreter. The system is built for theautomated conversion of ASL into textual content and speech.
In our project we basically focus on producing a model which can recognize Fingerspelling based hand gestures in order to form a complete word by combining each gesture.

II. REVIEW OF LITERATURE

Conversion of sign language to text is an important area of research as it can help bridge the communication gap between the hearing-impaired and the hearing population. Over the years, researches have explored various approaches and techniques to develop systems that can accurately convert sign language to text.
With the help of literature survey, we realized that the basic steps in handgesture recognition are:

- Data acquisition
- Data preprocessing
- Feature extraction
- Gesture classification

According to the paper on “Human Hand gesture Recognition using a Convolution Neural Network” by Hsien-1 Lin, Ming- Hsiang Hsu, and Wei-kai Chen (graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan), they have constructed a skin model to extract the hands out of an image and then apply binary threshold to the whole image. After obtaining the threshold imagethey calibrate it about the principle axis in order to center the image about the axis. 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 thismodel they produced an accuracy of around 95% for those 7 gestures.

III.METHODOLOGY

The system is a vision-based approach. All signs are represented with bare handsand so it eliminates the problem of using any artificial devices for interaction.
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 dataset. Steps we followed to create our dataset are as follows.
We used Open computer vision (opencv) library in order to produce our dataset.
Firstly, we captured around 100 images of each of the symbol in ASL(American Sign Language) for training purposes and around 100 images per symbol for testing purposes.
First, we capture each frame shown by the webcam of our machine. In each frame we define a Region of Interest(ROI) which is denoted by a blue bounded square .Then, we apply Gaussian Blur filter to our image which helps us extract various features of our image.

IV.WORKING MODEL

The system is a vision-based approach. All 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 800 images of each of the symbol in ASL (American Sign Language) for training purposes and around 200 images per symbol for testing purpose.
First, we capture each frame shown by the webcam of our machine. In each frame we define a Region Of Interest (ROI) which is denoted by a blue bounded square. Then, we apply Gaussian Blur Filter to our image which helps us extract various features of our image.

GESTURE CLASSIFICATION

The approach which we used for this project is : Our approach uses two layers of algorithm to predict the final symbol of the user.

Algorithm Layer 1:

1. Apply gaussian blur filter and threshold to the frame taken with open cv 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

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 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.

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).

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 us 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}

Finger spelling sentence formation

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 below.

Autocorrect Feature:

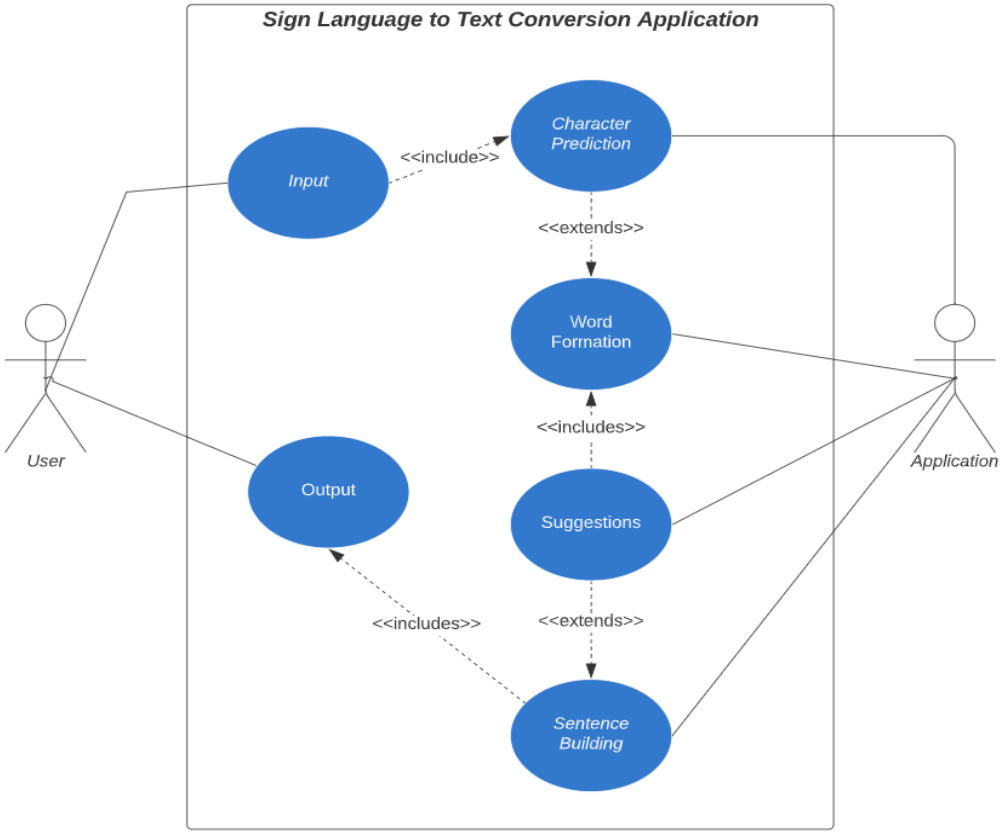
A python library **Hunspell_suggest** is used to suggest correct alternatives for each (incorrect) input word and we display a set of words matching the current word in which the user can select a word to append it to the current sentence .This helps in reducing mistakes committed in spellings and assists in predicting complex words.

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 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 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 labeled 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 labeled value and is zero exactly when it is equal to the labeled 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



V. MODULES

TensorFlow:

TensorFlow is an end-to-end open-source platform for Machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in Machine Learning and developers easily build and deploy Machine Learning powered applications.
TensorFlow offers multiple levels of abstractions so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.
If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution strategy API for distributed training on different hardware configurations without changing the model definition.

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 with images and text data easier.

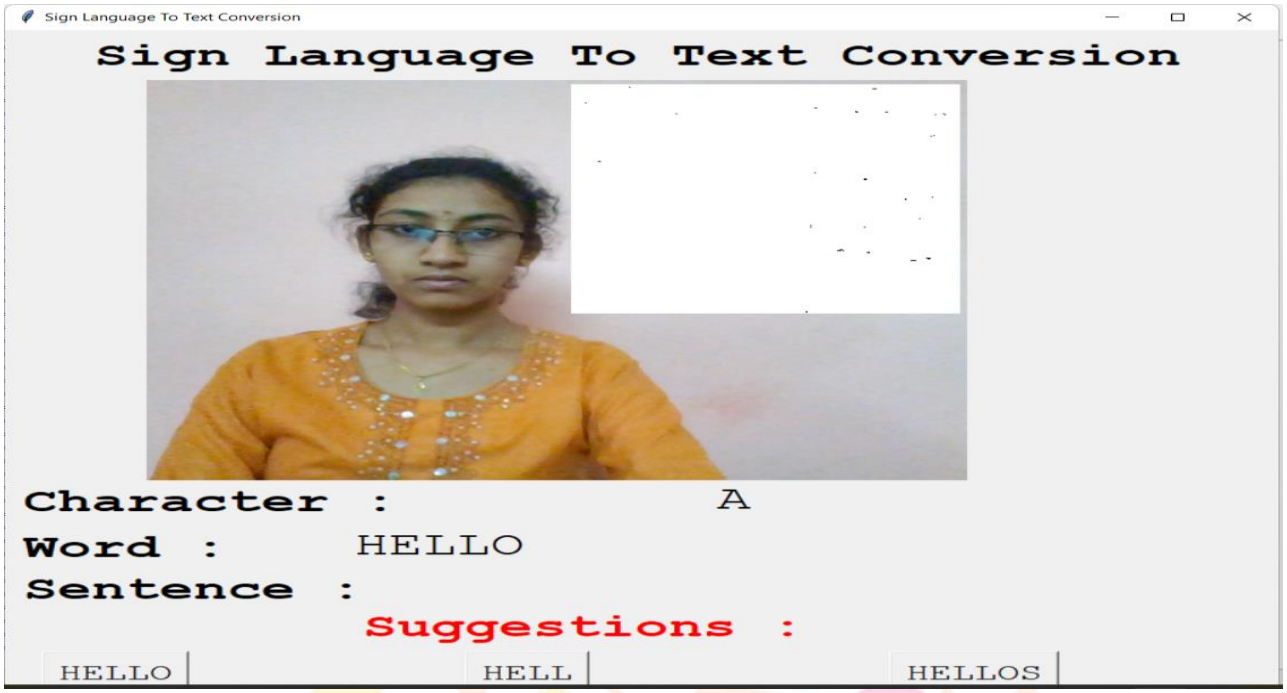
OpenCv:

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.

VI. TESTING

Software testing can begin as soon as there is executable software/program. Software testing is an examination used to offer information to stakeholders regarding the quality of the product or service being tested. Software testing can also give a corporation with an objective, unbiased picture of the software, allowing them to appreciate and comprehend the risks of software implementation. The practice of executing a program or application with the goal of detecting software bugs is one example of a test technique (errors or other defects). Software testing can give users and/or sponsors with objective, impartial information about the quality of software and the danger of it failing. When and how testing is undertaken, as well as the results, are typically determined by the entire approach to software testing or development.

VII. OUTPUTS:



VIII. CONCLUSION AND FUTURE SCOPE

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 98.0% on our dataset. We are able to 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 are able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.

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.

IX. 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.

[2] Ziaie, Thomas m uller, Mary Ellen Foster, and Alois Knoll "A Naïve Bayes Munich, Dept. of Informatics VI, Robotics and Embedded Systems, Boltzmannstr.3, DE-85748 Garching, Germany.

[3] https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[4] Mohammed Waleed Kalous, Machine recognition of Auslan signs usingPowerGloves: Towards large-lexicon recognition of sign language.

[5] aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural Networks-Part-2/

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

[7] 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 2014Lecture Notes in Computer Science, vol 8925. Springer, Cham