Sign Language Recognition and Text to Speech Conversion

1st Md. Kausar Islam Bidhan
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
md.kausar.islam.bidhan@g.bracu.ac.bd

2nd Sadia Yesmin
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
sadia.yesmin@g.bracu.ac.bd

3nd Shifat E Jahan
Senior Lecturar
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
sifat.jahan@g.bracu.ac.bd

4nd Annajiat Alim Rasel
Senior Lecturar
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
annajiat@g.bracu.ac.bd

***Abstract*—Since, generally people of vast majority do not understand sign language and it is difficult to find experienced interpreters nowadays, we tried to develop a system based real-time model on neural networks for American Sign Language (ASL) finger writing. In our method, the hand is first filtered to identify the category to which the hand gestures belong before being subjected to a classifier. Our approach yields a 95.7**

***Index Terms*—components, configuring, style, styling, insert**

I. Introduction

Out of all the sign languages currently in use, American sign language is the most widely used. Sign language is the only form of communication available to those with DM, who only have a communication impairment. The act of conveying information from one person to another through a variety of media, including words, gestures, body language, and pictures, is referred to as communication. Hand gestures are widely used by people who are deaf and mute (DM) to communicate. Gestures are used in nonverbal communication and can be visually interpreted. The deaf and dumb utilize sign language to communicate non-verbally. Our study's main objective is to develop a model that can recognize hand movements based on Fingerspelling and put them together to form a whole word.

II. Motivation

Due to the differences between sign language and standard text, normal people and DM persons cannot communicate with one another. As a result, they are only able to communicate through their eyes. Other people will be able to comprehend the movements easily if there is a standard interface that converts sign language into text. In order to enable DM individuals to communicate with one another without really comprehending one another's languages, research has been done on a vision-based interface system. The objective is to create user-friendly human-computer interfaces (HCI) that enable computers to comprehend sign language. There are numerous sign languages used around the world, including British Sign Language (BSL), Indian Sign Language (ISL), French Sign Language, and American Sign Language (ASL). The American Sign Language (ASL), the French Sign Language (FSL), the British Sign Language (BSL), the Indian Sign Language (ISL), the Japanese Sign Language (JSL), and other still-under-development sign languages are among the many sign languages spoken throughout the world.

III. Methodology

The ultimate goal determines how the system functions. Communication doesn't require the use of any technical intermediaries because every indication is made by hand.

A. The Data Set Generation

During the course of the project's development, we looked for pre-existing datasets but were unable to locate any raw image datasets that satisfied the needs of the task at hand. We could only find the datasets as RGB values. As a result, we went ahead and compiled our own set of numbers. The following are the procedures we used to compile our data set. In order to generate our dataset, we relied on the Open computer vision(OpenCV) library. To begin, we photographed roughly 800 examples of each ASL symbol for use in training and roughly 200 examples for use in testing. We begin by recording every single image that is displayed on our computer's webcam. We have outlined a ROI in blue in each of the images below to indicate where we want to focus our analysis. The following illustration shows how we take the

RGB original and extract the ROI before converting it into a grayscale image. The Gaussian Blur filter, which facilitates feature extraction, is then applied to the image to complete the process. Here is what the image looks like after being blurred using the gaussian method.

B. Gesture Classification

- ***Here is the approach we adopted for this project:*** Our method uses two levels of algorithms to forecast the user's final symbol.

- ***Algorithm Layer 1:***
  1) After feature extraction in opencv, the image is processed by first applying a gaussian blur filter and a threshold.
  2) The transformed image is fed into a convolutional neural network model for prediction.
- ***Algorithm Layer 2:***
  1) Many groups of signs are detected with consistent detection results.
  2) We then use classifiers developed specifically for these subsets to categorize data between them.
- ***Layer 1***
  The CNN Model:
  1) ***1st Convolutional Layer:*** The first convolution layer must cope with the 128x128 input image. The first convolutional layer uses the data with 32 filter out parameters (3x3 pixels respectively).
  2) ***1st Layer :*** The images are pooling down which means that the value in each 3x3 is retained. This results in a reduced resolution of 63x63 pixels for our image.
  3) ***2nd Convolutional Layer :*** The output of the first pooling layer, a 63-by-63 matrix, is used as input for the third layer, a convolutional neural network.
  4) ***2nd Pooling Layer :*** At the second pooling layer, the resulting images are downsampled once more, this time with a maximum pool size of 3x3. This brings the final image resolution down to 30x30.
  5) ***1st Densely Connected Layer :*** The first densely connected layer, which takes the images as input, is then given the output of the second convolutional layer, which is subsequently molded into an array of 30x30x32 = 28800 values. A 28800-valued array is fed into this layer. The second tightly connected layer receives the output from this layer. We used a dropout layer with a value of 0.5 to avoid overfitting.
  6) ***2nd Densely Connected Layer:*** The second densely connected layer, a completely connected 96-neuron network, receives the outputs from the first densely connected layer.
  7) ***Final layer :*** The final layer receives its input from the output of the second densely connected layer, and its number of neurons is equal to classes which were being classified (including the alphabets and blank symbols).
- ***The Activation Function :***
  Here, in each layers, we have implemented the Rectified Linear Unit (ReLU). The ReLu calculates max(x,0) for each input pixel. This contributes nonlinearity to the formula and assists in the learning of features that are more complex. By requiring less computing time, it contributes in eradicating the issue of disappearing gradients and expedites the training process.
- ***The Pooling Layer:***
  We have used the Max-Pooling algorithm on the picture that was provided to us, with the pool size set to (3, 3) and the ReLU Activation Function. This leads to a decrease in the overall number of parameters, which in turn leads to a decrease in the entire computing cost and a reduce in overfitting.
- ***Dropout Layers:***
  Overfitting, where the network's weights are so tuned to the training examples that they don't perform well with new examples. This layer zeroes a random set of activations in that layer. Even if some activations are removed, the network should classify or output a specific example[1].
- ***Dropout Layers:***
  Adam optimizer updates the model based on loss function output. Adam incorporates the advantages of the adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp) stochastic gradient descent method extensions.
- ***Layer 2***
  To get as close as we can to correctly identifying the symbol being displayed, we are employing a two-layer algorithm to verify and predict symbols that are most similar to each other. While conducting tests, we discovered that the following symbols were misdisplaying and giving other symbols instead:
  1) For U: R and D
  2) For D: U and R
  3) For S: T,U,K and S
  4) For I: M and N

  As a result, we developed three distinct classifiers to handle the aforementioned scenarios:
  1) U,R,D
  2) T,K,U,S
  3) I,M,N
- ***Formation of a finger-spelled sentence:***
  1) **The Implementation:**
     We demonstrate the letter and add it to the current string that exceeds a certain threshold. In that instance, the current dictionary, the number of detections of the current symbol, is cleaned to remove the potential of an incorrect letter . A blank is defined as an area with no pattern or texture. In any other scenario, it prints a space after the

word it thinks will be the last one, and the current sentence is appended to the one that comes after it.

2) **Autocorrect Feature :**
For each (incorrect) input word, a python module called Hunspell propose is used to recommend appropriate alternatives. The user is then presented with a list of alternatives that are compatible with the current word, from which he or she can select a word to add to the existing statement. If the user chooses a word, it will be added to the current sentence if it matches the existing term. Thus, fewer spelling mistakes are made, and it also makes it easier to anticipate challenging terms.

C. Training and Testing :

To filter out unwanted details, we apply a gaussian blur to our grayscale input images after converting them from RGB. The images are resized to 128x128, and an adaptive threshold is used to separate the hand from the background. After performing all of the aforementioned operations on the input images, we then feed them into our model for training and testing. The prediction layer guesses correctly which category the image falls within. This means that the output is scaled. We were able to do this in large part thanks to the softmax function. The output of the prediction layer will initially differ greatly from the true value. We have improved things by training the networks on labeled data. The output of the prediction layer will initially deviate greatly from the true value. The Cross Entropy is utilized as a performance statistic while categorizing data. It is a continuous function that is exactly zero unless it reaches its labeled value, at which point it is nonnegative. Accordingly, we made sure the cross-entropy was as close to zero as possible. We do this by modifying the neural networks' weights within the network layer. The cross entropy can be easily computed with a built-in function in TensorFlow. After determining the cross entropy function, we used Gradient Descent to fine-tune it; in particular, we used the Adam Optimizer.

IV. Challenges Faced :

In the course of this project, we encountered a number of obstacles. The lack of a complete dataset was the first challenge we encountered. Since it was much more manageable to work with only square images, we preferred dealing with raw images as CNN in Keras. We looked everywhere, but we couldn't find a suitable dataset, so we compiled one ourselves. Choosing which filter to employ on the photos to extract useful characteristics and utilize the generated image as input for the CNN model presented the second issue. Before settling on gaussian blur, we evaluated a number of filters, including binary threshold and canny edge detection.

V. Results :

Our model achieves an accuracy of 98.0% when layers 1 and 2 are combined, which is greater than the accuracy of the majority of the recent research articles on American Sign Language. Our model achieves an accuracy of 95.8% with just layer 1 of our method. The vast majority of these studies examine the effectiveness of hand detection hardware like the Kinect. In [2], the authors apply Kinect and convolutional neural networks to develop a system that can recognize Flemish sign language with an error rate of only 2.5%. It is shown in [3] that a recognition model can be constructed with a vocabulary of 30 words and an error rate of 10.90% using a hidden markov model classifier. Average accuracy for 41 static Japanese sign language gestures is 86%, as reported in [4]. Map [5], which used depth sensors, had an accuracy of 99.99% for signers that had already been detected and ranged from 83.58% to 85.49% for new signers. Their recognition software also made advantage of CNN. We don't use a background subtraction approach in our model, in contrast to some of the other ones that are described here. As soon as we attempt to use background subtraction in our project, there can be some errors. Despite the fact that the majority of the work discussed above uses Kinect sensors. Sensors like Kinect are so uncommon and prohibitively expensive for the great majority of potential customers, our model's use of a standard laptop's webcam is a tremendous advantage. Below you will find our confusion matrices.

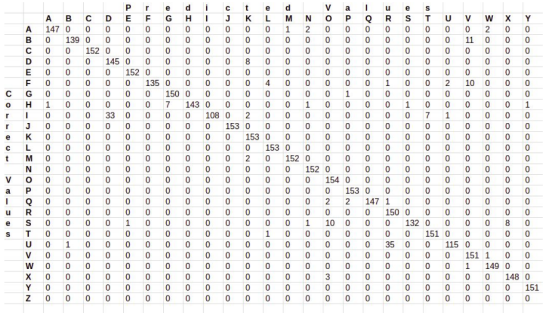

Fig. 1. **Algorithm 1**

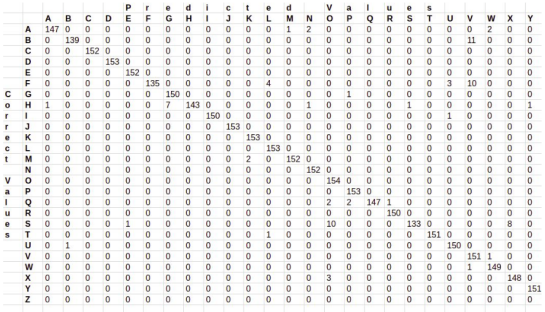

Fig. 2. **Algorithm 1 + Algorithm 2**

VI. Future Scope :

By experimenting with a variety of algorithms for background subtraction, we intend to increase the level of accuracy that can be achieved regardless of the complexity of the

background. Additionally, we are contemplating the possibility of enhancing the preprocessing in order to improve the accuracy with which gestures can be predicted under low-light conditions.

VII. Conclusion :

The purpose of this report is to describe the development of a functional real-time vision-based American Sign Language recognition system for DM people using ASL alphabets. On our dataset, we ended up achieving an accuracy rate of 98.0%. Following the implementation of two layers of algorithms, in which we verify and predict symbols which are more similar to one another, we were able to improve our ability to make accurate predictions. As long as they are accurately shown, there's no background noise, and the amount of light is adequate, we can recognize practically all of the symbols by acting in this way.

References

[1] Aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convol utional-Neural-Networks-Part-2/

[2] 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.

[3] 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).

[4] N. Mukai, N. Harada, and Y. Chang : Japanese Fingerspelling Recognition Based on Classification Tree and Machine Learning. Kyoto, Japan, 2017, pp. 19-24. doi:10.1109/NICOInt.2017.9.

[5] 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).