Volume: 12 Issue: V Month of publication: May 2024
DOI: https://doi.org/10.22214/ijraset.2024.60947

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

Conversion of Sign Language to Text

Mr. Kunal Singh[1], Mr. Mayank Shahi[2], Mr. Nikhil Gupta[3]

Abstract: Sign language is one of the oldest and most natural forms of language communication. These languages are essentially designed to support deaf people. However, since most people do not understand sign language and it is difficult to find an interpreter, we have developed a real-time method for fingerspelling using neural networks in sign language. In our approach, the hand first passes through the filter and, after applying the filter, passes through a classifier that predicts the gesture category. Our method provides higher accuracy for the 26 letters of the alphabet.

I. INTRODUCTION

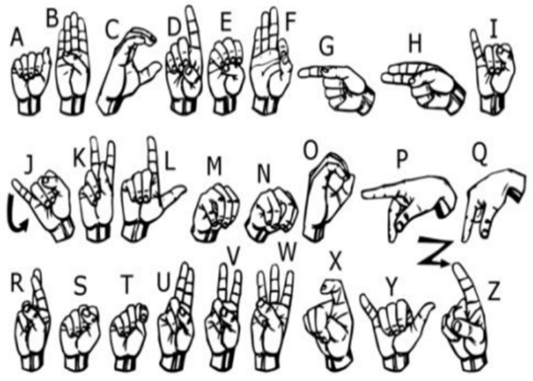
American Sign Language is a major sign language. Since the only impairment a D&M person has is related to communication, and they are unable to use spoken language, their only means of communication is sign language. Communication is the process of exchanging ideas and information through various means such as language, signals, behavior and visual elements. Deaf and mute people (D&M) use their hands to make various gestures to express themselves to others. Gestures are non-verbally exchanged messages that can be understood visually. This non-verbal communication between deaf people is called sign language.

II. LITERATURE SURVEY

In recent years, a great deal of study has been conducted on the recognition of sign language. After conducting a thorough literature analysis, we have determined that the basic steps of hand gesture identification are as follows:

- Data Acquisition
- Data Pre-processing
- Training
- Gesture classification

1) *Data Acquisition:* We collect hand motion data for each alphabet with a webcam and classify each gesture based on the alphabet that corresponds with it. The main challenge here is handling the massive variety of human hand look caused by a multitude of hand movements, different skin-color possibilities, and differences in view points, scales, and camera speed during scene capture.



2) *Data Pre-processing:* Processing takes longer, though, because the photos were taken in color. To decrease this, we need to convert those to black and white and remove every line using Gaussian blur and thresholding.
3) *Training:* The next step is to train the dataset to recognize the gestures. It is accomplished through the use of convolution neural networks, which are characterized through many layers.
4) *Prediction:* An intuitive Graphical User Interface (GUI) is essential. The system needs to produce correct text output by matching the motions it receives from the camera to the gestures in the trained dataset.

III. CONVOLUTION NEURAL NETWORK

Unlike standard neural networks, CNN's layers include neurons arranged in three dimensions: width, height, and depth. Rather than being totally linked to every other neuron in the layer, a layer's neurons will only be related to a limited fraction (window size) of the layer preceding it. Additionally, since we can condense the full-size image into a single vector of class scores by the time the CNN design is complete, the final output layer will have dimensions (number of classes).

A. Convolution Layer:

In the convolution layer, we employ a small window size that typically measures 5 by 5 and extends to the depth of the input matrix. Learnable filters with window-sized dimensions comprise the layer. We calculate the dot product of the input values at a specific location for each iteration and then move the window by a stride size, typically 1. The response of the matrix at each spatial position will be provided by a 2-Dimensional activation matrix that will be generated as we proceed. Put differently, the network will identify filters that activate when specific types of visual elements, such a splotch of a specific color or an edge with a specific orientation, are detected.

B. Pooling Layer:

We use a pooling layer to reduce the size of the activation matrix and, ultimately, the learnable parameters. There are two types of pooling:

1) *Max Pooling:* This method only takes the top four values and utilizes a window size, like a 2*2 window. When we finally have an activation matrix that is half the size it was initially, we'll close this window and continue.
2) *Average Pooling:* In this technique, all values are averaged over a given period of time.

C. Fully Connected Layer:

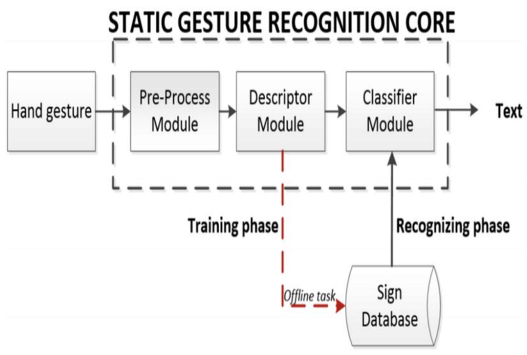
Neurons in a convolution layer are only connected to a local region; in a fully connected region, all inputs are coupled to all neurons.

D. Final Output Layer:

The final layer of neurons, which has a count equal to the entire number of classes, is connected to the fully connected layer's values to estimate the likelihood that each image would belong to a different class.

IV. METHODOLOGY

The method used by the system is vision-based. To avoid the issue of utilizing any artificial gadgets for interaction, the majority of the signs are portrayed with just the hands.



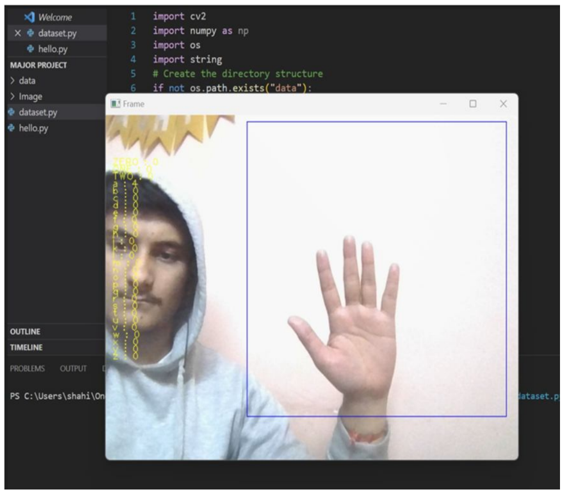
A. Data Set Generation:

We looked for pre-made datasets for the project, but we were unable to locate any that met our needs for raw image format. The datasets are limited to RGB values alone.

We thus made the decision to produce our own data set. The following are the steps we used to construct our data set.

We created our dataset using the Open Computer Vision (OpenCV) framework. Initially, we took about 1000 pictures of every ASL symbol for training purposes, and then we took about 200 pictures of each symbol for testing.

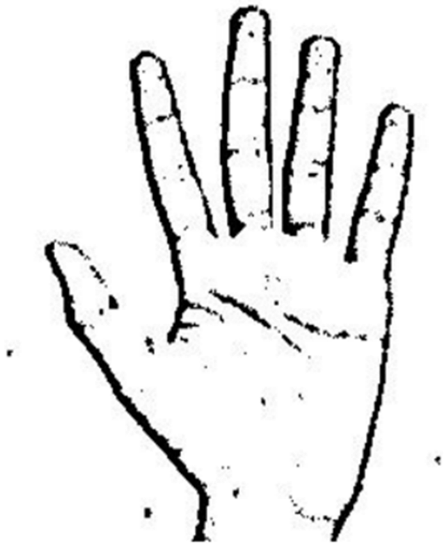
Initially, we record every frame that our machine's webcam produces.
We designate a region of interest (ROI) in each frame, apply RGB, thresholding, and Gaussian blur as shown in the image below:



```
import cv2
import numpy as np
import os
import string
# Create the directory structure
if not os.path.exists("data"):
```

```
PS C:\Users\shahi\On
```

Gray Scale image



Gaussian filtered image

V. ALGORITHM

A. *Algorithm Layer 1:*

1) After feature extraction, apply the Gaussian blur filter and threshold to the OpenCV-captured frame to obtain the processed image.
2) After this image has been processed, it is fed into the CNN model for prediction. If a letter is identified in more than 50 frames, it is printed and used to build the word.
3) The blank sign is used to represent the space between words.

B. *Algorithm Layer 2:*

1) We identify several symbol sets that yield comparable detection results.
2) Next, we use classifiers designed specifically for those sets to classify between those sets.

a) Layer 1:

CNN Model :

- *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.
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.

- *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.
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.

- *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.

- *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.

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).

b) 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}

➢ *Activation Function:*

Rectified Linear Units, or ReLus, are employed in every layer (even fully linked and convolutional neurons). Max(x,0) is computed by ReLu for every input pixel.

This enhances the formula's nonlinearity and facilitates the learning of increasingly complex features. It assists in solving the vanishing gradient issue and shortens the computation time, which speeds up training.

c) Pooling Layer:

Using a relu activation function and a pool size of (2, 2), we perform Max pooling to the input image.

This lowers the number of parameters, which lowers the cost of computing and lessens overfitting.

➢ *Dropout Layers:*

Overfitting is a problem in which, following training, the network's weights are so adjusted to the training instances that it performs poorly when presented with fresh examples. By setting them to zero, this layer "drops out" a randomly selected subset of activations in that layer. Even if certain activations are dropped, the network should still be able to offer the correct categorization or output for a given case.

➢ *Optimizer:*

To update the model in response to the loss function's output, we employed the Adam optimizer. Adam combines the benefits of two extensions—adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp)—of two stochastic gradient descent techniques.

VI. CONCLUSION

A functional real-time vision based system for D&M individuals has been created to recognize American sign language for the ASL alphabets. Our dataset yielded a final accuracy of 98.0%. By utilizing two tiers of algorithms, we may enhance our prediction by validating and forecasting symbols that exhibit greater similarity to one another. As long as the symbols are displayed correctly, there is no background noise, and the illumination is sufficient, we can identify practically all of the symbols in this manner.

VII. FUTURE SCOPE

Our goal is to increase accuracy even while dealing with intricate backdrops by experimenting with different background subtraction algorithms. Additionally, we are considering enhancing the preprocessing to more accurately anticipate gestures in low light.

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] 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.

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

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

[5] aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convol utional-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 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham

[8] 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) 25

[9] 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
[10] 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)
[11] https://opencv.org/
[12] https://en.wikipedia.org/wiki/TensorFlow
[13] https://en.wikipedia.org/wiki/Convolutional_neural_network