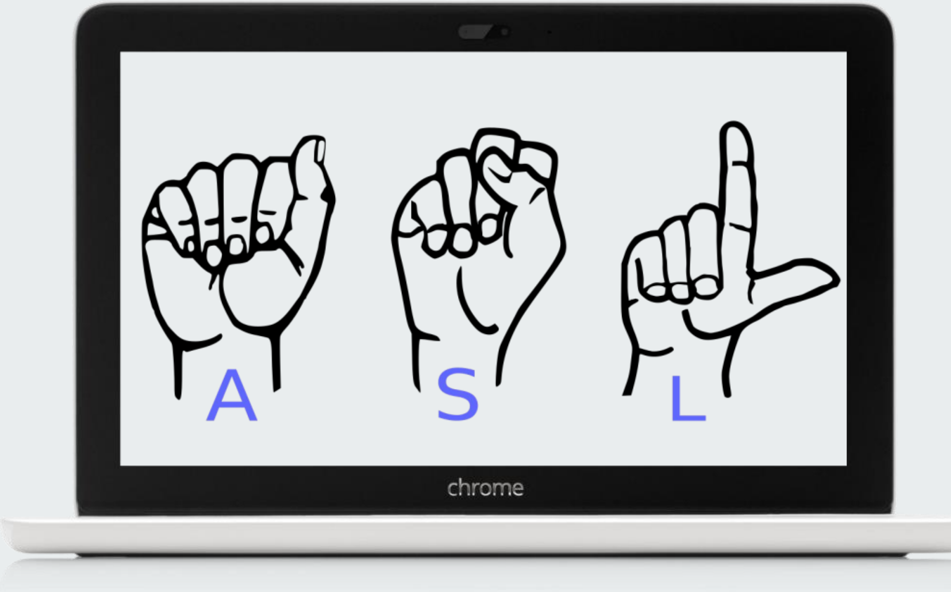
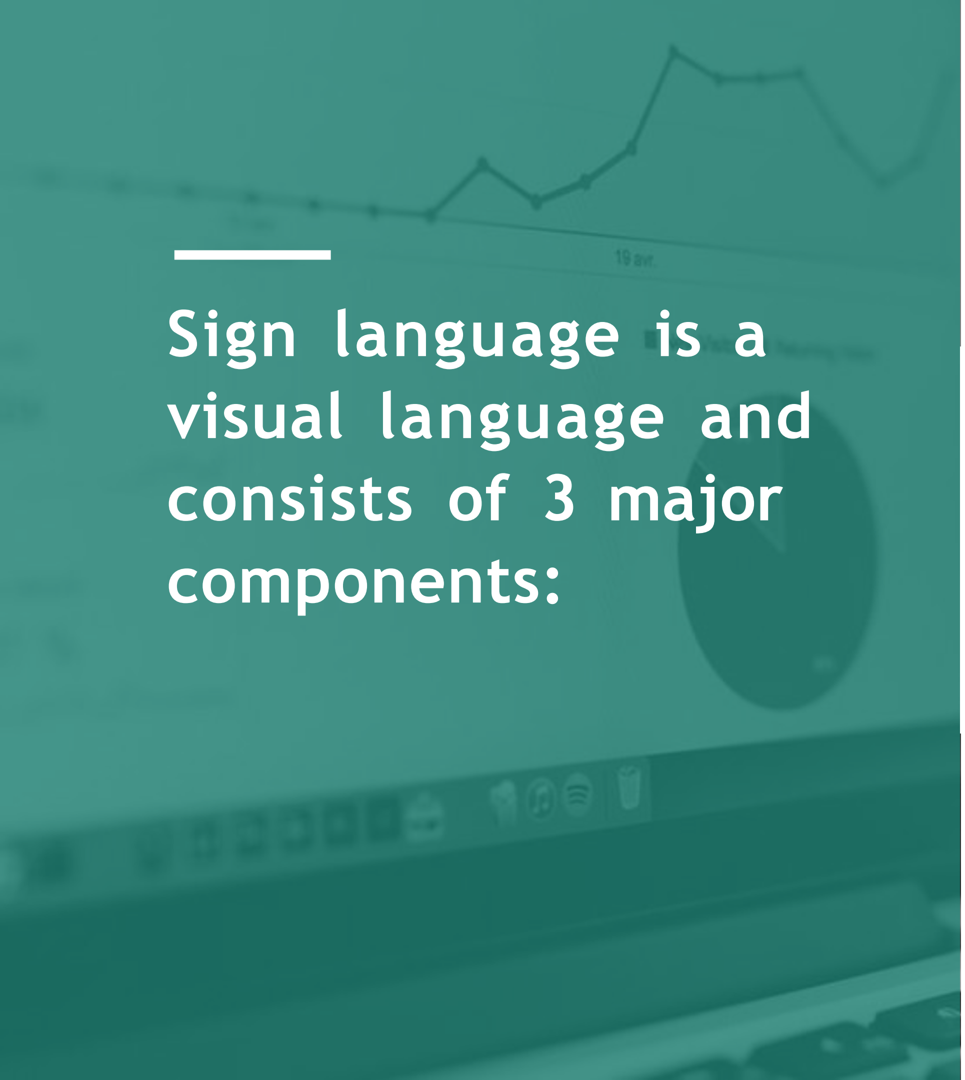
Conversion of Sign Language to Text

For Dumb and Deaf

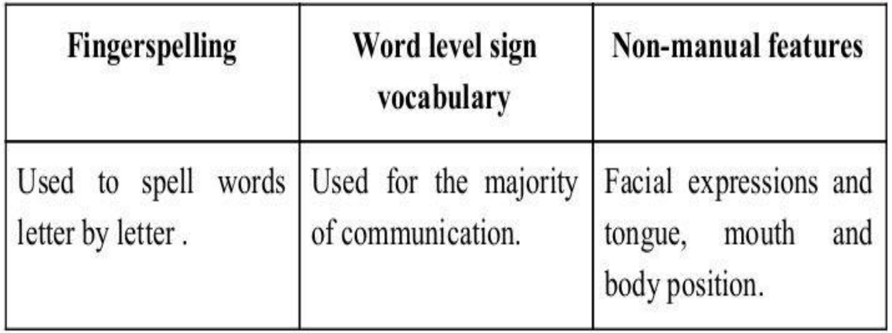


Abstract

Our project aims to create a computer application and train a model which when shown a real time video of hand gestures of American Sign Language shows the output for that particular sign in text format on the screen.

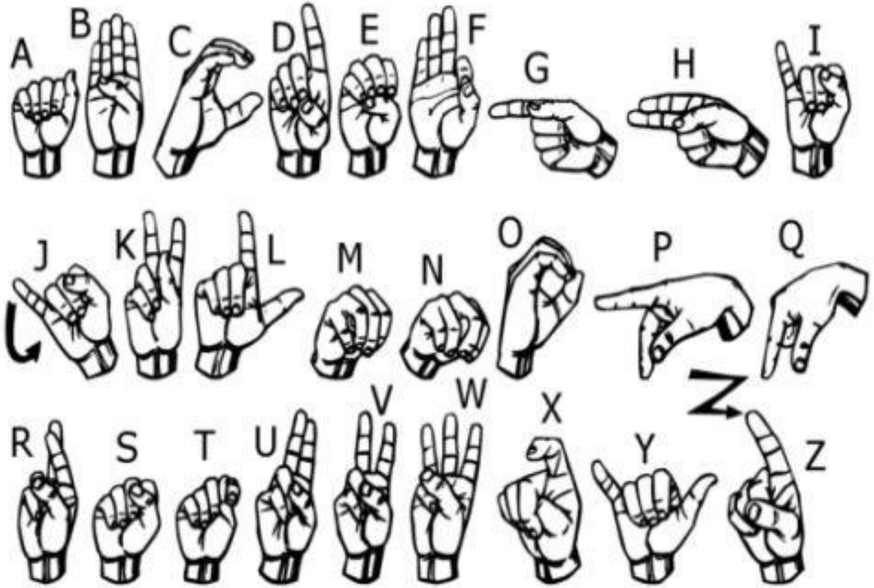


Sign language is a visual language and consists of 3 major components:



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.

We implemented 27 symbols(A-Z, blank) of ASL in our project.



Methodology

How we generated data set and did Data Preprocessing ?

Gray Scale Image

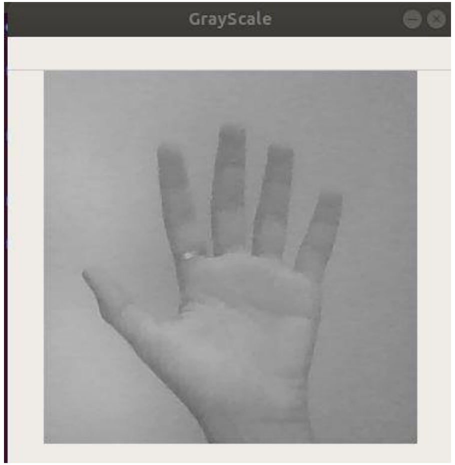
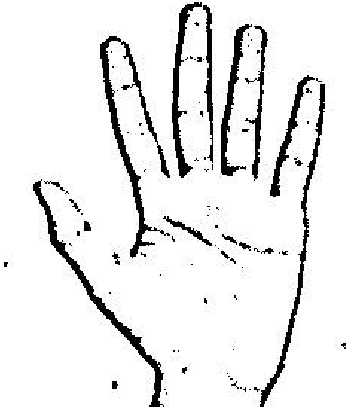


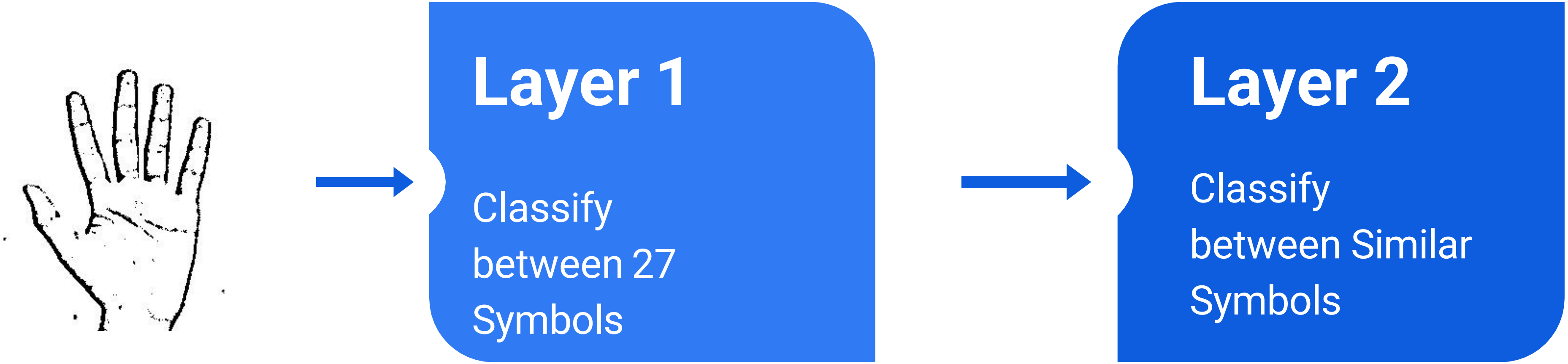
Image Post Gaussian Blur



Why we Created our own Dataset ?

- In our project, we attempted to locate pre-existing datasets but were unsuccessful in finding raw image datasets that aligned with our specific criteria. The datasets we came across were primarily composed of RGB values.
- Our search for already compiled datasets in the form of raw images, matching our project requirements, proved fruitless. The available datasets were predominantly structured around RGB values.
- Despite our efforts to discover suitable datasets, we were unable to find raw image datasets that met our criteria. Existing datasets were predominantly presented in the format of RGB values.

Gesture Classification



Layer 1

Classify between 27 Symbols

Layer 2

Classify between Similar Symbols

Algorithm Layer 1:

- Apply a Gaussian blur filter and threshold to the OpenCV-captured frame to obtain a processed image post feature extraction.
- Pass the processed image through the CNN model for prediction purposes.
- If a letter is consistently detected for more than 50 frames, acknowledge and print the identified letter.
- Consider the detected letter in the process of forming a word.
- Recognize the blank symbol to account for spaces between words in the final output.

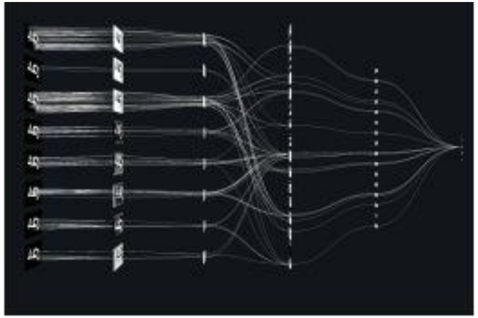
Algorithm Layer 2:

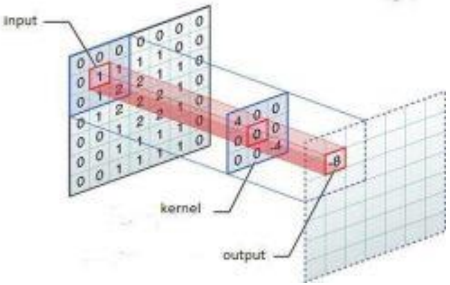
- Identifying multiple sets of symbols reveals similar outcomes upon detection.
- To address this, specialized classifiers are created for individual sets, facilitating classification among them.
- During testing, certain symbols were observed to exhibit inaccuracies in detection, resulting in the misclassification of other symbols.
- Specifically, the following symbols were found to be problematic, displaying issues in proper recognition and often leading to the incorrect identification of alternative symbols.

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

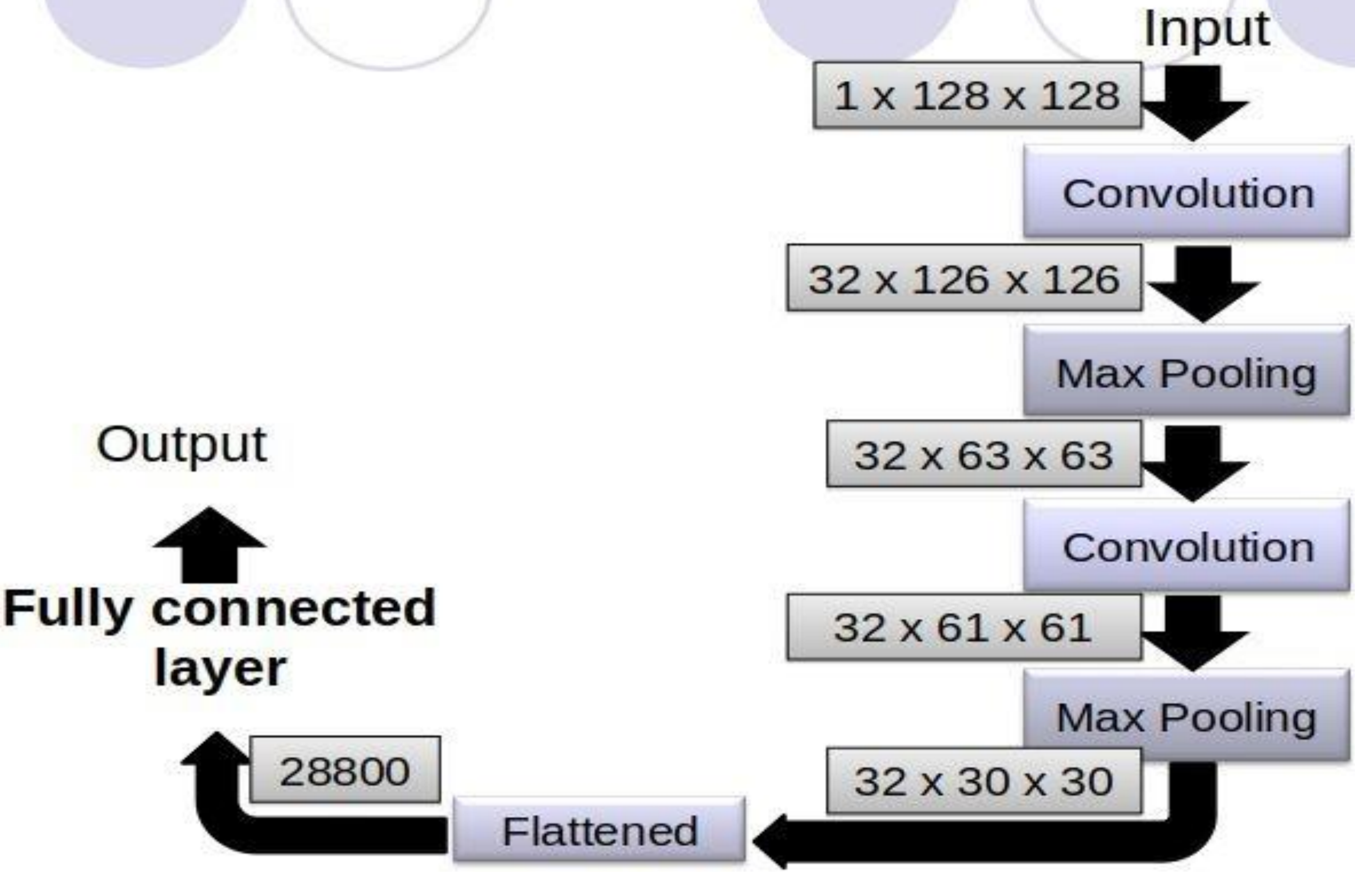
Convolutional Neural Networks

- CNNs consist of multiple convolutional layers each layer containing numerous “filters” which perform feature extraction.
- Initially these “filters” are random and by training, the feature extraction gets better by better.
- It’s primarily used for image classification.





Our CNN Classifier Model


Input
1 x 128 x 128
Convolution
32 x 126 x 126
Max Pooling
32 x 63 x 63
Convolution
32 x 61 x 61
Max Pooling
32 x 30 x 30
Flattened
28800
Fully connected layer
Output

Finger Spelling Sentence Formation

Implementation

- If the count of a detected letter surpasses a predefined threshold (set at 50 in our code) and no other letter is in close proximity within a specified threshold (20 in our implementation), we print the identified letter and include it in the current string.
- To prevent the possibility of an incorrect letter prediction, we clear the current dictionary containing the count of detections for the present symbol.
- When the count of a detected blank symbol (representing plain background) exceeds a specific threshold and the current buffer is empty, no spaces are detected.
- In the case where the buffer is not empty, it predicts the end of a word by printing a space, and the current buffer content is 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.

Challenges Faced

- ➢ We couldn’t find a dataset with raw images of all the asl characters so we made our own dataset.
- ➢ Second issue was to select a filter for feature extraction. We tried various filter including binary threshold, canny edge detection, gaussian blur etc. ,of which gaussian blur filter was giving better results.
- ➢ Issues were faced relating to the accuracy of the model we trained in earlier phases which we eventually improved by increasing the input image size and also by improving the dataset.

Software Requirements

- Python 3.6.6
- Tensorflow 1.11.0
- OpenCV 3.4.3.18
- NumPy 1.15.3
- Matplotlib 3.0.0
- Hunspell 2.0.2
- Keras 2.2.1
- PIL 5.3.0

Limitations of our model

- The model works well only in good lighting conditions.

- Plain background is needed for the model to detect with accuracy.

Conclusion

- 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 an accuracy of **95.7%** on our dataset.
- Prediction has been improved after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other.

Future Scope

- ❖ 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.

Project Under Open Track

Bolt 2.0

Efforts by

Shreyash Manindra- 21BIT0440

O Revanth- 22BIT0623

K Vaibhav-21BIT0579

Thank You !