“LINGUISTIC SIGN TRANSLATOR”

BACHELOR OF TECHNOLOGY

IN

(COMPUTER SCIENCE & ENGINEERING)

3RD YEAR (5TH SEM)

SESSION (2022-2023)

ASHOKA INSTITUTE OF TECHNOLOGY & MANGEMENT

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SUBMITTED BY -

SHREYASKAR JAISWAL (2006410100055)

AINDREE ANIL (2006410100003)

AMAN SINGH (2006410100008)

RITU PARNA BANERJEE (2006410100042)

SUBMITTED TO -

DR. PRITI KUMARI

CERTIFICATE

Session (2022-2023)

This is to certify that the work incorporated in the project report entitled Linguistic Sign Translator is a record of work carried out by our team of **Aman Singh, Aindree Anil, Shreyaskar Jaiswal and Ritu Parna Banerjee**. Under my guidance and supervision for the award of Bachelor of Technology in the faculty of Department of Computer Science & Engineering from Ashoka Institute of Technology & Management, Varanasi.

To the best of my knowledge and belief the project report follows the orders of the university and meets the requirements.

SIGNATURE –

Dr. Priti Kumari

DECLARATION

We hereby declare that the project entitled **“Linguistic Sign Translator”** Submitted by us in the partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in the faculty of Department of Computer Science & Engineering** from Ashoka Institute of Technology & Management, Varanasi of Dr A.P.J. Abdul Kalam Technical University is record of our own work carried under the supervision and guidance **.** To the best of our knowledge in this project have not been submitted to any other University or Institute for the award of degree.

Signature :
Name : Shreyaskar Jaiswal
Roll No. : 2006410100055
Date :

Signature :
Name : Aindree Anil
Roll No. : 2006410100003
Date :

Signature :
Name : Aman Singh
Roll No. : 2006410100008
Date :

Signature :
Name : Ritu Parna Banerjee
Roll No. : 2006410100042
Date :

ACKNOWLEDGEMENT

I would like to express my deep gratitude to **Dr. Priti Kumari (Professor & HOD),** my research supervisors, for their patient guidance, enthusiastic encouragement, and useful critiques of this research work.

I would also like to thank our respected faculty for their advice and assistance in keeping my progress on schedule. My grateful thanks are also extended to **Mr. Ankur Srivastava (Assistant Professors)** and other professors for their help in doing the data analysis, and helped me calculate different types of algorithms and for their support in the dataset's measurement.

I would also like to extend my thanks to the technicians of the laboratory of the CSE department for their help in offering me the resources in running the program.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

CONTENTS

1. ABSTRACT: -

Sign language is an extremely important communication tool for many deaf and mute people. So, we proposed a model to recognize sign gestures using different Machine Learning Algorithms. We utilized a Pre-Trained SSD Mobile net V2 architecture trained on our own dataset in order to apply Transfer learning to the task. We developed a robust model that consistently classifies Sign language in majority of cases.

Additionally, this strategy will be extremely beneficial to sign language learners in terms of practising sign language.

Various human-computer interface methodologies for posture recognition were explored and assessed during the project. A series of image processing techniques with Human movement classification was identified as the best approach. The system can recognize selected Sign Language signs with the accuracy of 70-80% without a controlled background with small light.

2. INTRODUCTION: -

American Sign Language (ASL) is a language that maps hand gestures to alphabetical letters. In ASL, for each letter, there exists a special "fingerspelling". To give an example, the letter "L" is indicated by stretching the index finger of the right hand straight up and orientating the thumb towards left. Eventually, when viewed from the front, the shape of the hand would resemble the letter "L" (see Figure 1).

Emerged back in 1817 to educate deaf people, today, ASL has around 1.000.000 disabled users. Yet, the number of non-disabled users is relatively low, which is approximately 250.000 [1].

To promote its use, in this project, we are investigating ways of interpreting ASL with the aid of machine learning and image processing principles. More precisely, given a live video stream (through the camera of the computer), by examining the position of hands, we aim to identify which letters are described according to ASL fingerspelling set.

Each recognized letter is to be projected onto a window to inform the user simultaneously. In this way, the user will be able to write words onto the screen in an interactive manner.

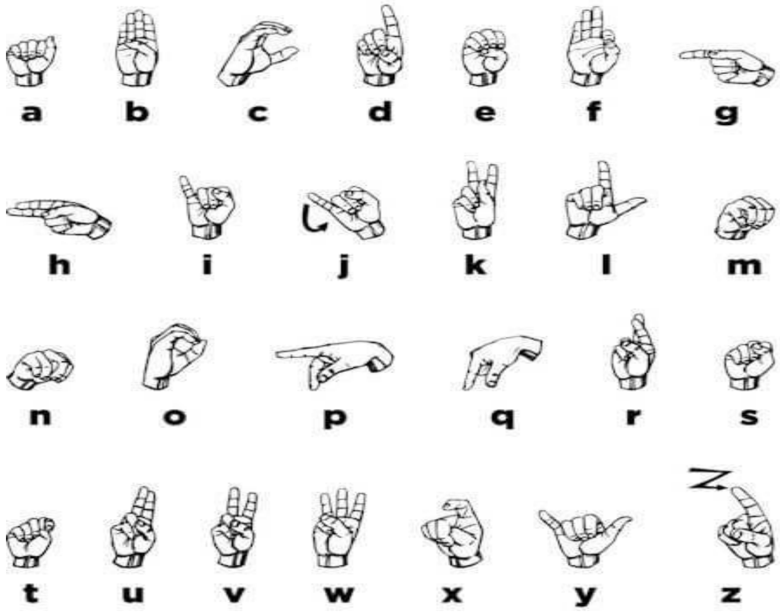


Figure 1: Hand depictions of letters of the alphabet in American Sign Language (ASL).

Before the implementation, we did research to identify which machine learning algorithms may work best for sign language recognition. According to our findings, we have concluded that the following ones are likely to serve our purposes:

- Linear Discriminant Analysis
- Convolutional Neural Networks
- Support Vector Machine
- Gaussian Mixture Modelling
- Linear Predictive Coding
- K Nearest Neighbours

Similarly, to identify how we could represent images as feature vectors, we needed to identify some basic image processing techniques. According to our research, we have come up with the following ones:

- Static Thresholding
- Adaptive Thresholding
- Colour Thresholding
- Gaussian Filters
- Morphological Operations (Dilation and Erosion)
- Logical Operations (AND, OR, NOT, XOR)
- Edge Detection
- Histogram Equalization

3. SYSTEM REQUIREMENTS: -

SOFTWARE:

- Visual Studio Code
- Scikit-Learn
- Open CV
- Datasets
- Internet Connection

HARDWARE:

- Web Camera
- Processor: Up to 3Gb RAM
- Graphics Card (optional)

4. DATASET DESCRIPTION: -

In our project proposal, we mentioned that we would use the dataset available on https://www.kaggle.com/emresulun/sign-language-letter-images. However, when we examined the samples in the dataset, we realized that they are not suitable for us to use. More precisely, as can be seen in Figure 2; the images in the dataset are not of the same dimensions, some of them contain faces, and there are many items in the background, which altogether complicate their pre-processing.

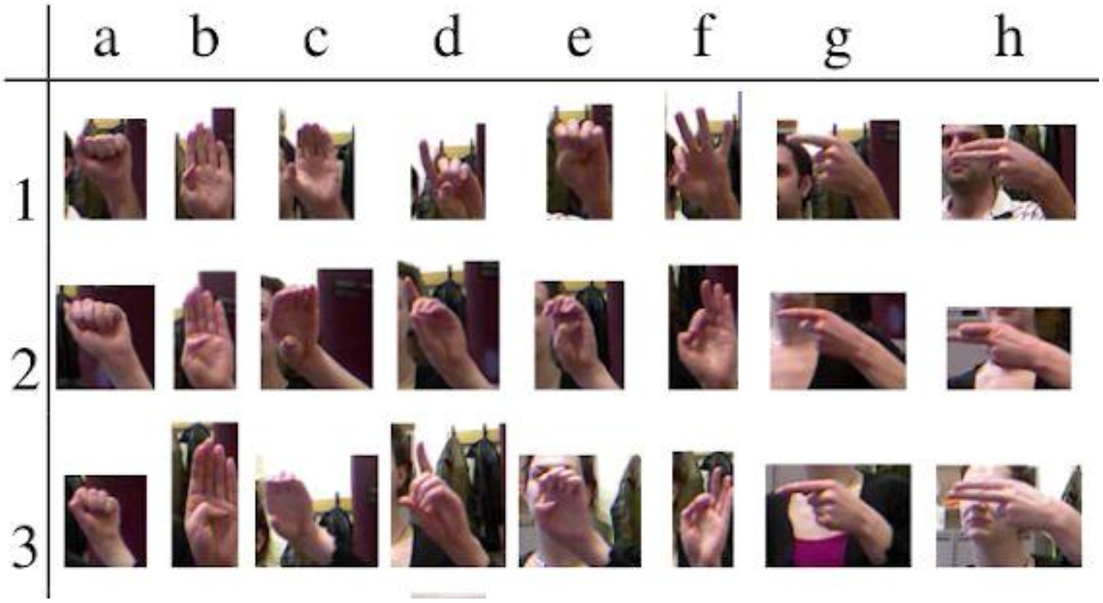


Figure 2: Some of the samples taken from the dataset we *previously* proposed to use.

Consequently, we have decided to come up with our own dataset. To do so, we captured images of our hands via the webcam of our computers by periodically sampling images. The current distribution of our samples is given in Table 1. As it can be observed from the table.

	A	**B**	**C**	**D**
Number of samples	265	265	265	265

Table 1: Current (incomplete) distribution of all (training and test) samples

We created sample images of equal numbers so that our dataset will not be biased towards a specific label. Notice that the dataset is yet to develop, meaning that we aim to reach at least 1000 samples per each letter as we progress. Depending on the results, to get better results with Convolutional Neural Networks (CNN) algorithm, we may need to extend our dataset even further.

5. IMPLEMENTATION: -

Image Processing

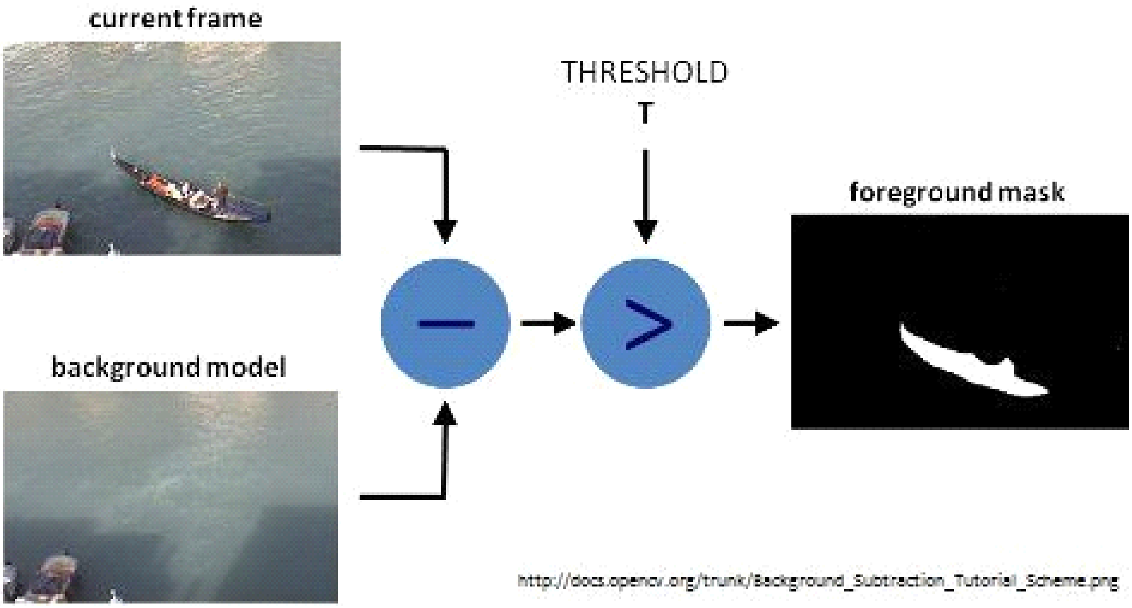
Regarding the image processing part, in order to extract hand shapes from the images, we have tried several approaches:

- Firstly, we have experimented with Gaussian filters and adaptive thresholding using different configurations. After a lot of trials, we have realized that background objects were present in the resulting images. Consequently, since we could not extract hand shapes only, we have discarded this approach.

- Secondly, we tried exploiting the "green screen" concept, which is used in films. This way, we could extract the outer contours of the hand properly; however, the inner details were lost. We thought that, in some cases, we may need to understand the inner details of the hand (like the orientation of fingers). Therefore, we had to disregard this method as well.

- Thirdly, hoping to get better results, we tried to use skin colour filtering [2] for detecting the hand posture from the Region of Interest (ROI). To implement it, we used HSV (Hue Saturation Value) colour modelling and range function of OpenCV together with some morphological operations of dilation and erosion. The results were considerably better to compare to the previous approaches that we have used; however, it gets problematic again in terms of the colour of the objects in the background so that in case of having objects with colours which enter to the range of specified lower and upper skin colour values, the program detects these objects (some parts of them) as the part of the vector image too. Basically, background clutter occurs. Other than that, it is an effective method for good results.

- Finally, we experimented using another method for recognizing the object from the complex backgrounds. The method that we tried was based on background subtraction [3]. When the program starts to execute, we let the program figure out the background by having a weighted average of 60 frames of it. After having an idea about the background, the program can understand when the new entry (our hand) enters the ROI and we could obtain this foreground model (again our hand) from the specified background by calculating the absolute difference between them. After the subtraction, we applied a simple threshold on the difference and we got our result.



http://docs.opencv.org/trunk/Background_Subtraction_Tutorial_Scheme.png

Figure 3: How background subtraction works.

- Since the field of Computer Vision has lots of challenges, we also did not pass away by solving all the problems applying this last technique; having an unstable background (even change of light) corrupts the accuracy of the segmentation of hand posture from the video, so a stable background is necessary for applying this method.

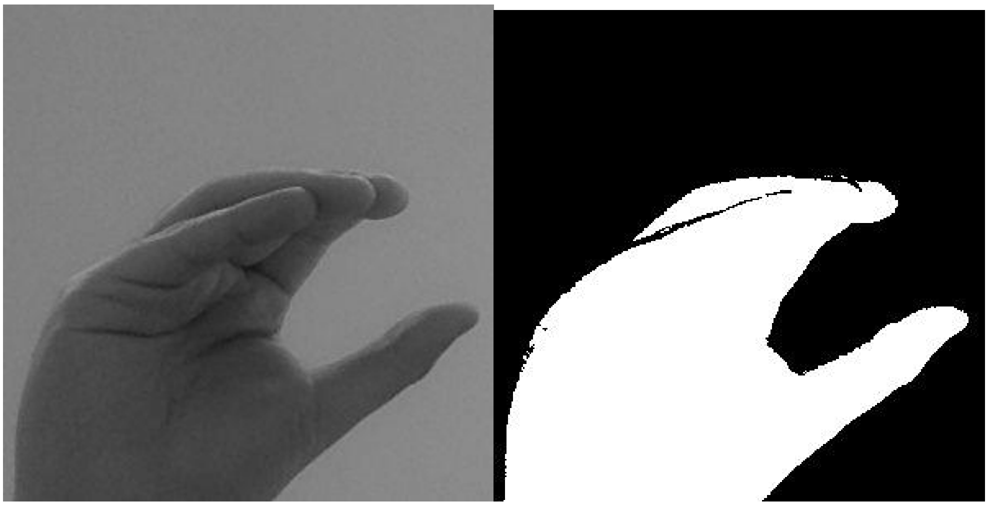


Figure 4: An example picture of the letter C, before (left) and after pre-processing (right).

Figure 4 shows a pair of images for the letter “C”. The one on the left is the grayscale image taken via the webcam from the video stream. The one on the right is its pre-processed version. All the pictures in our dataset are stored as pre-processed (binary) images.

Machine Learning

We have implemented 3 algorithms: LDA (Linear Discriminant Analysis), PCA (Principal Component Analysis) and SVM (Support Vector Machine). We used PCA to reduce features, then we used these features in LDA and SVM algorithms. We used scikit-learn library to implement these algorithms. Note that our current dataset has 5 different letters: A, B, C, D, and E. Their details (i.e., distribution in terms of training and test samples) are given in the following table.

	A	**B**	**C**	**D**
Train	250	250	250	250
Test	15	15	15	15

Table 2: Number of pictures for each letter divided into train and test data.

The accuracies of the algorithms we used are given below,

	LDA (Linear Discriminant Analysis)	**SVM (Support Vector Machine)**	**LDA + PCA (Principal Component Analysis)**	**SVM + PCA (Principal Component Analysis)**
Accuracy (%)	87%	97%	53%	60%

Table 3: Accuracies of different algorithms

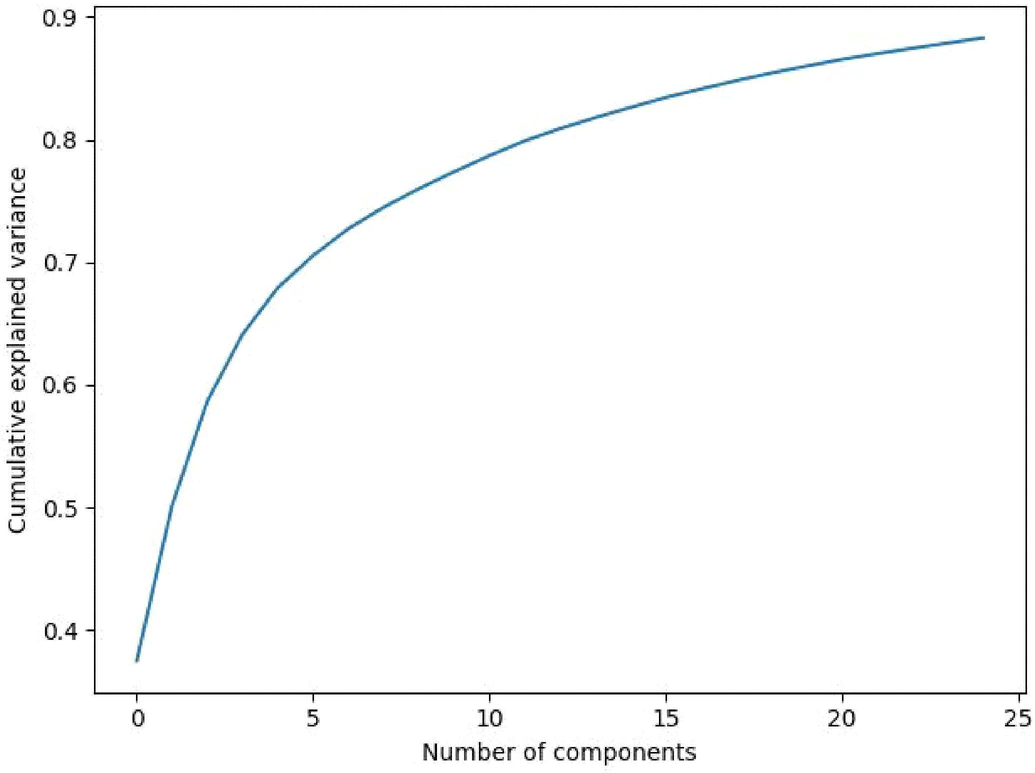


Figure 5: PCA number of component vs cumulative variance plot

We are using our pre-processed data - binary images which contain the pixels with values either 0 or 255 (black or white) to train the models. Since we are using these binary pixel values as our feature vectors, there could be some non-linear dependencies [4] among the feature values, which could be the reason why having less accurate results with PCA feature reduction (see Figure 5 and Table 3).

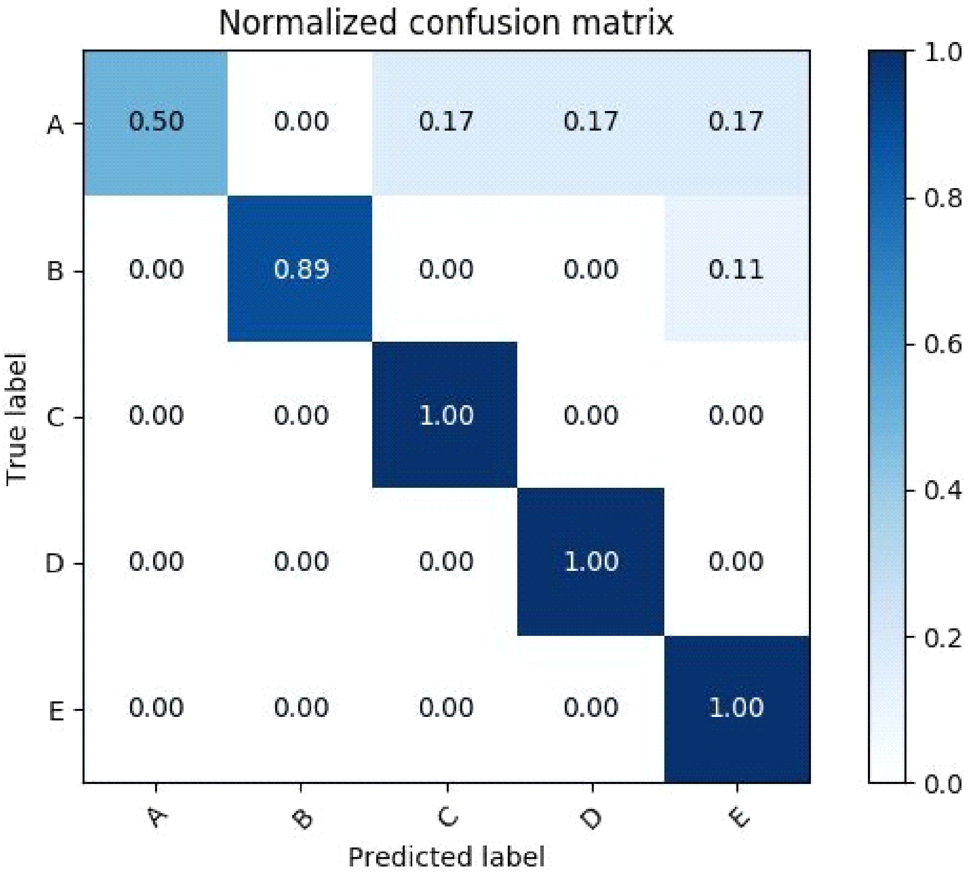
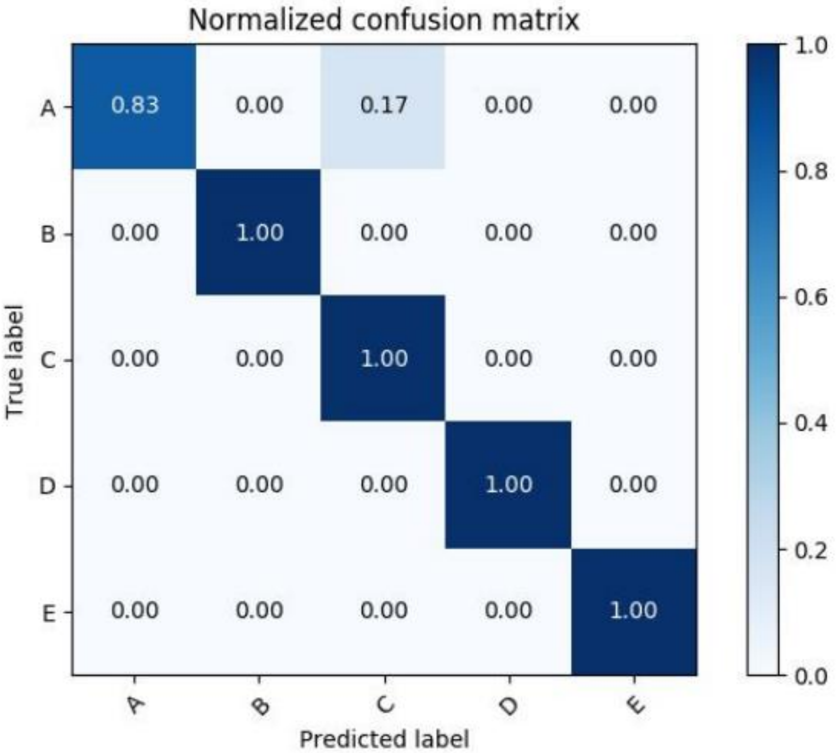


Figure 6: Normalized confusion matrix of LDA algorithm

In Figure 6, for LDA, our confusion matrix plot shows the true labels and the predicted labels of our 5 classes which are the first 5 letters of the alphabet. The intersection of the labels shows whether our prediction is true or not. For example, all the true labels' "C" are predicted correctly, thus the normalized intersection value is 1.00. However, it can be seen from column C, not all the predicted C values were C. One of the predicted values of C actually has true label A, so this decreased the normalized intersection value of predicted label A x true label A.

But the intersection between predicted label A and true label A is 0.50 since out of 6 true values of A, only 3 of them were predicted to be A and 1 was predicted to be C, 1 was predicted to be D and another one was predicted to be E while the true value was A.



On the other hand, for SVM, our test results are shown in the confusion matrix above (Figure 7). As already mentioned in Table 3, SVM results in more accurate predictions than LDA. The reason may be that SVM, when used with Radial Basis Function (RBF) kernel (to measure the similarity between the features in a non-linear - radial way), can handle the non-linearity of features while LDA cannot. In other words, features resulting from images can be better separated with a non-linear model and, thus, SVM with RBF kernel makes more accurate predictions.

In addition to these 3 algorithms, we started to implement CNN that is a powerful algorithm for image classification. CNN consists of three steps: convolution, polling, and flattening. Now, we implemented convolution step and we will continue for the next steps later.

6. REMAINING WORK: -

As the remaining part of the project implementation, we are planning to:

- Complete the implementation of the CNN algorithm.
- Implement our model for the project using the different machine learning algorithms (Mel Frequency Cepstral Coefficients Analysis, Gaussian Mixture Modelling, Linear Predictive Coding, K Nearest Neighbours) that we are supposed to test.
- Adjust the values for thresholding, morphological and logical operations and apply new techniques such as Feature and Edge Detection for better pre-processing of images.

7. DIVISION OF WORK AMONG TEAMMATES: -

Since **Aman Singh** and **Aindree Anil** take Image Processing classes, they are responsible for pre-processing our dataset (transformation of webcam images to binary images and, then, to feature matrices).

Shreyaskar Jaiswal and **Ritu Parna Banerjee** are responsible for implementing, and testing predictive analytic models i.e., different machine learning models (LDA, SVM, CNN) and algorithms (PCA).

All the team members are involved in the report phase.

8. REFERENCES: -

- R. Mitchell, T. Young, B. Bachleda and M. Karchmer, "How Many People Use ASL in the United States? Why Estimates Need Updating", *Sign Language Studies*, vol. 6, no. 3, pp. 306-335, 2006. Available: 10.1353/sls.2006.0019.
- A. Rosebrock, "Tutorial: Skin Detection Example using Python and OpenCV", *PyImage Search*, 2019. [Online]. Available: https://www.pyimagesearch.com/2014/08/18/skin-detection-step-step-example-usi ng-python-OpenCV/.
- W. classifier? and D. Antenucci, "What can cause PCA to worsen results of a classifier?", *Cross Validated*, 2019. [Online]. Available: https://stats.stackexchange.com/questions/52773/what-can-cause-pca-to-worsen-r results-of-a-classifier.