Gesture Recognition Using Deep Learning

Kunal Chhabria
School of Information Technology and Engineering
Vellore Institute of Technology, Vellore,India
kchhabri@sfu.ca

V.Priya
School of Information Technology and Engineering
Vellore Institute of Technology, Vellore,India
vpriyacse@vit.ac.in

I.Sumaiya Thaseen
School of Information Technology and Engineering
Vellore Institute of Technology, Vellore,India
isumaiyathaseen@vit.ac.in

2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) 978-1-7281-4142-8/20/$31.00 ©2020 IEEE 10.1109/ic-ETITE47903.2020.135

***Abstract*— Deep Learning is a revolution that has potential to cause tremendous technological upheavals across all walks of humanity. Here I present an application to translate alphabets of Indian Sign Language in real time and a pipeline to make similar applications. The main steps include Data set Generation, Model Training, Frame by Frame Classification. The application works in real time and in varying backgrounds and the user can type alphabets by doing corresponding gestures in front of webcam. While some other implementations require several expensive devices that user might need to wear, my application only needs a camera.**

Keywords*— Deep Learning, Gesture Translation, Indian Sign Language, Transfer Learning.

I. INTRODUCTION (*HEADING 1*)

Deep Learning and computer vision are some of the most prominent and futuristic technologies of today. Gesture recognition is a very significant application of these technologies. Automated Sign Language translation is a problem that can leverage gesture recognition. The users would be able to communicate without having a translator and this can impact a lot of specially abled people.

An important application of gesture recognition is in the field of Augmented Reality and Virtual Reality. Using gesture recognition users can carry out tasks without the need of keyboards or voice. The user should only perform hand gestures and the computer will understand what action user is trying to perform. While voice control is a good way of hands-free control, it brings with along significant delay while gesture recognition can happen almost instantaneously. Also, voice control requires much more effort in AR/VR applications and gesture control feel much more intuitive and natural.

Gesture recognition also brings with it touchless user interfaces. The users can interact with devices without the need of touch. Imagine, public installations of computer such as libraries or airports etc. where thousands of people interact with computer to get information. In such places many people would not want to touch the screens and having option of gesture control would be fantastic. In addition, gesture control can also be used in home automation and drone control, robot control etc.

One of the most important applications of gesture control is in the field of gaming. X-box and Nintendo Wi already have their implementations of gesture control. Finally gesture control is really cool and tempting to use. We have all seen iron man movies and how Tony Stark interacts with his AI Jarvis. Such technologies are not far from reality and gesture control is an essential part of it.

.

II. LITERATURE ON NERUAL NETWORKS

This survey [1] shows a comparison of the best CNN for practical application and also how fast they are. Residual networks Residual Networks [2] are currently the state of art neural networks for image classification. Different versions of Resnet can be used but since this is a real time application resnet18 works best as it is the lightest network but is also quite effective. An effective approach is presented [3] for deviceless gesture recognition but training and testing times are too long. Hence, a 2d CNN model is used which works reasonably well and is also fast. The authors [4] analyze various transfer learning techniques and where they can be useful. They can be applied effectively for areas like image classification and detection. As a result, my problem can greatly benefit from transfer learning. It will be applied by using a resnet18 model and training by unfreezing last few layers and using differential learning rates.

III.METHODOLOGY

In order to do gesture translation: perform classification on each frame and if same class is recognized for last 20/30 frames, type that alphabet.

A. Gesture Classification

Gesture classification has following main steps.

- Data Creation
- Data Augmentation
- Training the model on GPU
- Inference in CPU

For data Creation multiple videos of each gesture are shot in varying backgrounds and angles. From these videos every 5th frame is extracted. The final dataset contains 500 images for each alphabet. Random Zoom, Random Rotate, Contrast change, Blurring were used for data Augmentation.

Fig1. Sample of dataset

The pretrained resnet18 model was finetuned with differential learning rates on Nvidia Tesla K80 GPU and it takes less than an hour to obtain 99% accuracy and inference takes less than a second. I used the fantastic fast.ai framework made by Jeremy Howard for this task.

B.Real time Gesture Translation

To do Gesture Translation each frame is classified and if same gesture is classified in last 20/30 frames type it. In order to perform translation in different backgrounds it is necessary to extract background from foreground. There are multiple ways to do this such as Active contours method, Image segmentation, Largest Contours, Background Subtraction. I considered two real time situations: camera is still, camera in motion.

When the camera is still background subtraction works flawlessly and efficiently.

It can be seen in the image that the gesture is extracted perfectly. A grayscale version of dataset was generated to train the model for background subtraction.

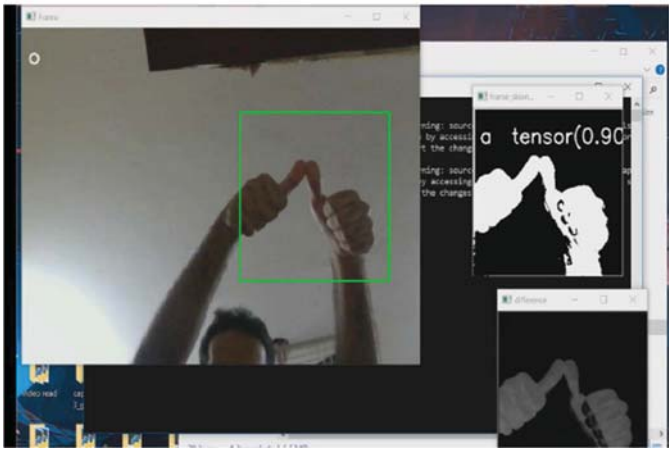

Fig 2. Background subtraction

The working of background subtraction is as follows:

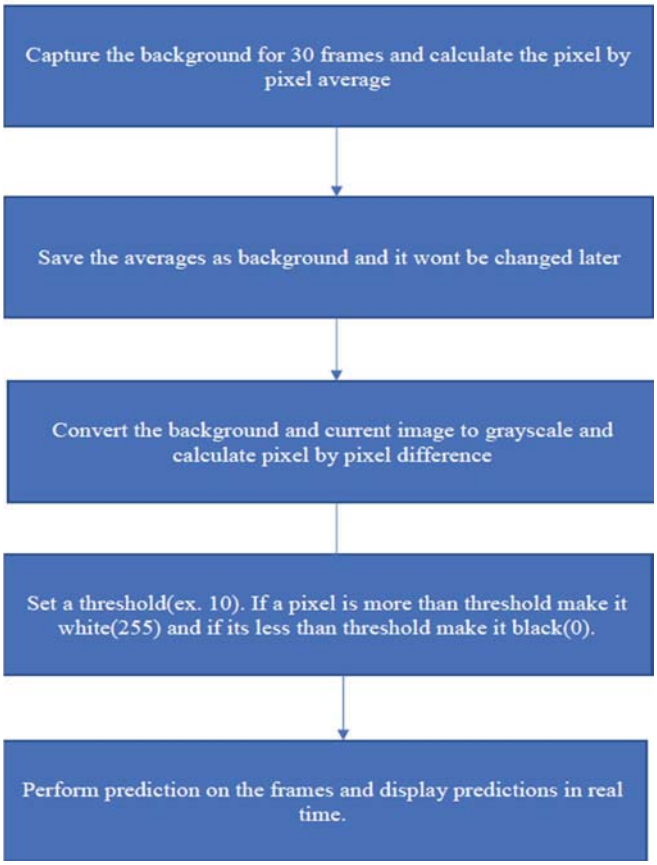

Fig 3. Background subtraction working

Gesture Translation become much trickier when the camera is in motion or we have varying background.

Largest contour method might or might not work depending on how zoomed in the and is and how noisy the background is. Image segmentation using U-net might work if an annotated dataset can be generated but it still will have lower fps. Hence, I used skin color extraction to do the task. But the main challenge is that skin color range is different for everyone and if the range is too wide or too narrow subject will not be extracted efficiently. Also, same skin might have different HSV values in different lighting conditions. Hence, a skin sample must be captured to

initialize the range every time the application is used for a different person or lighting. Also, if the background is similar to user's skin color, accuracy will go down.

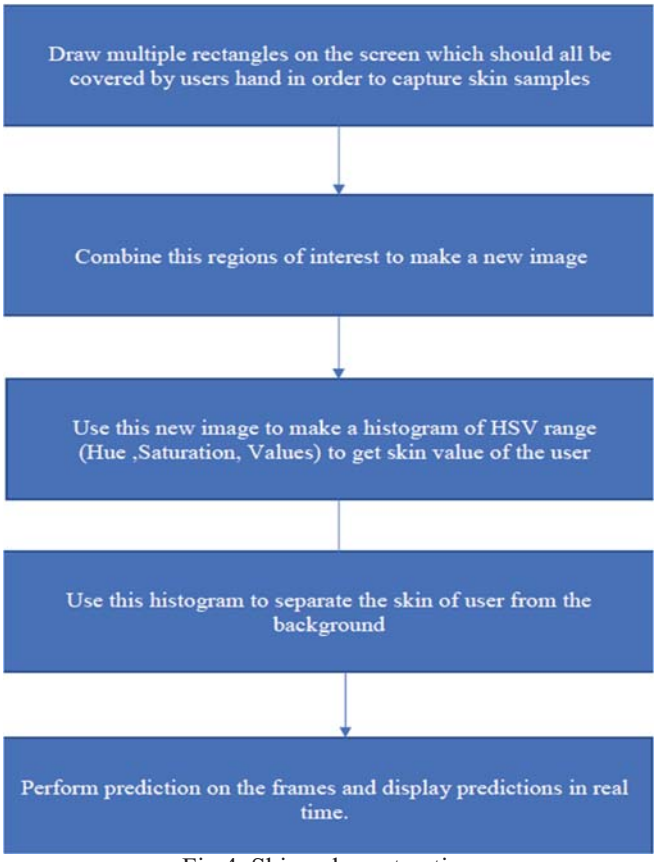

Fig 4. Skin color extraction

So the user has to cover the boxes with skin to get the color range.

Fig 5. Collecting skin samples

The results were good but there was room for improvement. The limitation with above method is that the sample of skin collected isn't big enough. So, to rectify that I am collecting skin samples 4 times. Two time for the front of hand and two times for the back of hand. While collecting skin samples boxes of 10 by 10 pixels are drawn. Larger boxes result in more outliers. A total
of 36 boxes and hence 3600 pixels are captured. As the sample size increases the problem of outliers occur. A few black or white pixels can skew the data. So to rectify that I am plotting bar charts and scatter plots of the skin data. The way lower and upper values of skin is detected is as follows.

Lower Value= Mean – 2*standard deviation
Upper Value = Mean+ 2*standard deviation

The lower and upper values are displayed to the user along with the bar charts. I used HSV color range throughout the project.

Average skin color would be in the range as shown.

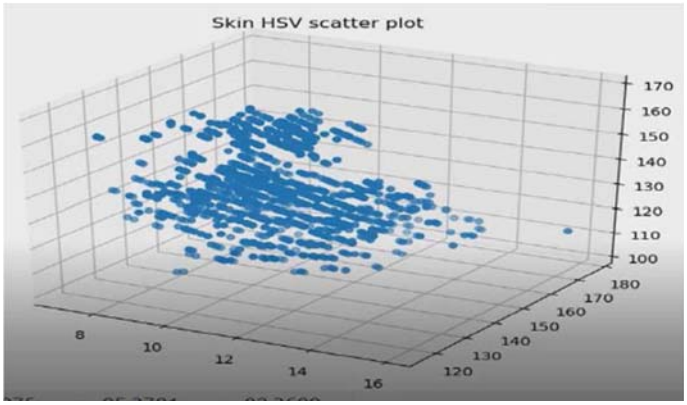

Fig 6: Skin HSV scatter plot

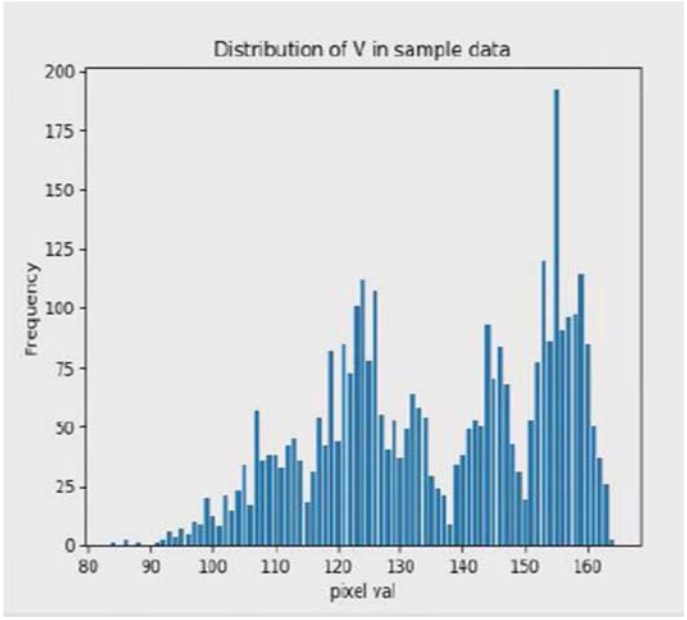

Fig 7. Skin color range

The output is shown below.

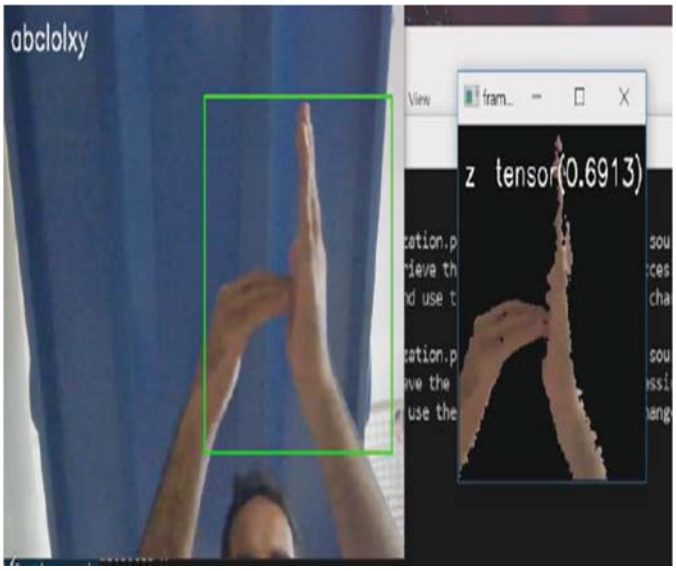

Fig 7. Output with skin color extraction

IV. Conclusion

The aim of this paper is to build a system to achieve real time gesture recognition with Indian Sign Language alphabets as the use case. In order to do so I created multiple versions of datasets, trained corresponding models with high accuracy and different methods based on camera and background settings. The project has resulted in a pipeline which can be used to train any gesture recognition application as only the dataset has to be changed and other steps remain the same.

REFERENCES

1. Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." arXiv preprint arXiv:1605.07678 (2016).

2. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

3. Molchanov, Pavlo, et al. "Hand gesture recognition with 3D convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2015.

4. Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.