SIGN LANGUAGE DETECTION USING MACHINE LEARNING

Mr. Rushikesh Bhalerao
Head of Department
rushikesh.bhalerao@pravra.in
Department of Information Technology

Mr. Piyush Bhujbal
bhujbalpiyush@gmail.com
Department of Information Technology

Mr.Shreyas Waghchaure
shreyaswaghchaure@gmail.com
Department of Information Technology

Mr. Rutik Malave
rutikmalave14@gmail.com
Department of Information Technology

Mr. Krushna Chavan
krushnachavan9545@gmail.com
Department of Information Technology

Sir Visvesvaraya Institute of Technology
A/p :Chincholi, Tal.:Sinnar,Dist.:Nashik,Maharashtra,India-422102.

Abstract- The dumb and deaf people have the most common problem of communication with the normal people.The physically challenged people who are disabled to talk and hear need to find a way of communication medium through which they can express their thoughts and solve their problems. Thus, to work upon it some experts designed a Sign Language. This language is different for different countries and then implemented it upon performing experiments over long times. The Indian Sign Language is probably the topic of study and interest. The thing is because of its simplicity and accuracy anyone can understand what the handicapped is trying to say.

In the above review we a team of four students under the guidance of our project guide are just giving a rough improvement and adding some automation techniques with Machine Learning and Computer Vision through which we are attempting to create the Indian Sign Language Detection algorithm through Deep Neural Networks. The main aim of this project is to create a simple, easy and ready to use application that predicts the symbol shown by the user in front of the camera and then the ML model in the backend predicts the output and shows it live on the screen.

Keywords: , ***Skin Segmentation, SVM, CNN,KNN,.***

1. INTRODUCTION

Communication is very important to human beings, as it enables us to express ourselves. We communicate through the speech, gestures, body language, reading, writing. However, for the speaking and hearing impaired minority, there is a communication gap. Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an emergency. Sign Language chiefly uses manual communication to convey meaning. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts.

Sign language is the standard form of communication among the speech and hearing impaired. The region-wise division of the sign language helps the users to have a facile method to convey information. As the larger population of society does not understand sign language. the speech, and hearing impaired usually rely on the human translator. The availability and affordability of using a human interpreter might not be possible all the time. The best substitute would be an automated translator system that can read and interpret sign language and convert it into an understandable form. This translator would reduce the communication gap that exists among people in society.

The SLR should be trained with many sign language data and its grammar for a smooth and uninterrupted sign language conversion. Every gesture created so far has a specific meaning and an application. Every sign language used all over the world is rich in grammar and vocabulary. SLR can be considered as a modified HC model, where the system can read and process the hands' movement. Such models would pave a path for barrier-free communication.

This can be very helpful for the deaf and dumb people in communicating with others as knowing sign language is not something that is common to all, moreover, this can be extended to creating automatic editors, where the person can easily write by just their hand gestures.

Sign Language is a form of communication used primarily by people hard of hearing or deaf. This type of gesture-based language allows people to convey ideas and thoughts easily overcoming the barriers caused by difficulties from hearing issues.

2. LITERATURE REVIEW

Tanuj Bohra an Indian based Canadian student persuing his graduation proposed a real- time two-way sign language communication system built using image processing, deep learning and computer vision. Techniques such as hand detection, skin color segmentation, median blur and contour detection are performed on images in the dataset for better results. CNN model trained with a large dataset for 40 classes and was able to predict 17600 test images in 14 seconds with an accuracy of 99% .Lionel Pigou and et al. [2] used two networks, consisting of three layers of convolution, each followed by max-pooling. One of the CNNs was trained to capture features from hand and the other from the upper body. The outputs of the two CNNs were concatenated and fed into a fully-connected layer. They utilized the dataset from CLAP14 [3] consisting of 20 Italian gestures by 27 subjects. They used both the depth and color images. They achieved an accuracy of 91.7%on cross-validation containing different users with different backgrounds from the training set and testing accuracy of 95.68% but it contained users and backgrounds from the training set. Alina K. and et al. [4] used a multi-layered Random Forest model, for the dataset collected using the Open NI + NITE framework on Microsoft Kinect. They achieved an accuracy of 87% for subjects on whom the system was trained whereas 57% for a new subject. Lementec and et.al [5] used glove-based motion tracking sensor to identify gestures. However, given the limitations of having such delicate sensors and gloves makes it unfeasible to use. Hussain and et.al [6] used VGG-16 architecture (CNN) to train and classify hand-gestures. But, in this case the hand gesture must be placed in front of a constant background and any deviation in background will result in incorrect classification. Yamashita and et.al [7] proposed a deep convolutional neural network model for classifying hand-gestures. But they were only able to classify 6 hand-gestures with around 88.78% accuracy. Pei Xum [8], captured raw RGB images using monocular camera, and used background subtraction techniques whereby achieving a classification of over 99% while classifying 16 gestures. B. Liao and et.al [9], used depth sensing camera and Hough transform for hand segmentation and CNN for training. They achieved an accuracy of 99.4% in classifying 24 gestures from the American Sign Language system (ASL).

3. AIMS AND OBJECTIVES

1. The Main objective of the project is to give an interface where deaf and non-deaf people can easily understand sign language through a graphical context.
2. The purpose of this project is to provide an efficient and accurate way to convert sign language into Graphical context Using Machine learning and flutter we look forward to create An Application that help both deaf and non-deaf people to communicate with each other.

4. PROPOSED SYSTEM

The goal of this project was to build a neural network able to classify which letter of the American Sign Language (ASL) alphabet is being signed, given an image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written and oral language. Such a translator would greatly lower the barrier for many deaf and mute individuals to be able to better communicate with others in day to day interactions. This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exists in higher rates among the deaf population, especially when they are immersed in a hearing world [1]. Large barriers that profoundly affect life quality stem from the communication disconnect between the deaf and the hearing. Some examples are information deprivation, limitation of social connections, and difficulty integrating in society Most research implementations for this task have used depth maps generated by depth camera and high resolution images. The objective of this project was to see if neural networks are able to classify signed ASL letters using simple images of hands taken with a personal device such as a laptop webcam. This is in alignment with the motivation as this would make a future implementation of a real time ASL-to-oral/written language translator practical in an everyday situation.

5. SYSTEM ARCHITECTURE

Sign language recognition is a topic which has been addressed multiple times and is not new. Over the last few years, different classifiers have been applied to solve this problem including linear classifiers, neural networks and Bayesian networks. Linear models are easy to work with, but require complex feature extraction for increased Accuracy. Work done by Singha and Dasn helped them achieve accuracy of 96% on 10 classes for images of gestures of one hand using Karhunen-Loeve Transforms Real- time American Sign Language Recognition with Convolution Neural Networks [1]. These transformations rotate the input frame feed and set up a new coordinate system based on the variance of the data. This is preceded by applying image pre-processing to extract the

significant part of the images. They use a classical linear classifier to identify fingers pointing in directions.Work done by Sharma uses a combination
of Support Vector Machines and k-Nearest Neighbours to recognize the symbol after background subtraction and noise removal[2]. They make use of contour tracing,
which represents hand contours. This system attained an accuracy of 61%.Bayesian networks like Hidden Markov Models have also achieved high accuracies [3]. These capture temporal patternsaccurately, but they require clearly defined models that are defined prior to learning.
Starner and Pentland achieved a highly accurate system (99%) using a Hidden Markov Model though the high accuracy was due to the usage of a sensored glove[4]. The glove helped them acquire accurate 3D spatial details. Neural networks have also been used to achieve Sign Language Translation. Neural networks have an
Advantage, which however, comes with a trade-off. They improve the accuracy by determining which model to make use of but this comes with the cost of compromising slower speed of training and usage. To date, most have been relatively shallow. Admasu and Raimond classified Ethiopian Sign Language correctly in 98.5% of cases using a feed-forward Neural Network [5]. They use a significant amount of image preprocessing, including image size normalization, image background
Subtraction, contrast adjustment, and image segmentation. Admasu and Raymond extracted features with a Gabor Filter and Principal Component Analysis. L.Pigou used a Microsoft Kinect to identify full-body features of a person showing the gesture [6].

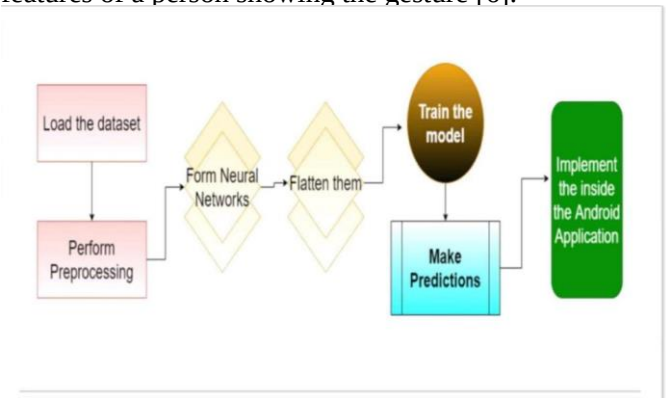


Fig. System Architecture Of Sign Language Detection

The open-source framework,TensorFlow object detection API makes it easy to develop, train and deploy an object detection model. They have their framework called the TensorFlow detection model zoo which offers various models for detection that have been pre-trained on the COCO 2017 dataset. The pre-trained TensorFlow model that is being used is SSD MobileNet v2 320x320. The SSD MobileNet v2 Object detection model is combined with the FPN-lite feature extractor, shared box predictor, and focal loss with training images scaled to 320x320. Pipeline configuration, i.e., the configuration of the pre-trained model is set up and then updated for transfer learning to train it by the created dataset. For configuration,dependencies like TensorFlow, config_util, pipeline_pb2, and text format have been imported. The major update that has been done is to change the number of classes which is initially 90 to 26, the number of signs (alphabets) that the model will be trained on.

6. IMPLEMENTATION

The implementation part of the project is broken into the following steps:
1. Data Preprocessing and Inference.
2. Model Selection and Training.
3. Exporting the trained model.
4. Integrating the TensorFlow lite model with Flutter.
5. Incorporating the Predictions into the Applications.
6. Testing and Optimization

Data Preprocessing and Inference

As part of the integration, we implemented the necessary data preprocessing steps within our Flutter application. This involved transforming the input data into the format expected by the Machine Learning model, such as scaling, normalization, or encoding. Once the input data was prepared, we utilized the TensorFlow Lite APIs to perform the model inference. The output of the inference provided us with the desired predictions or insights from the Machine Learning model. Activities for preprocessing done are:

1. Image dataset collection.
2. Image visualization using OpenCV.
3. Class creation for labelling.
4. Image Preprocessing using OpenCV, Keras and TensorFlow

Model Selection and Training

Before proceeding with the integration, we carefully selected an appropriate Machine Learning model that suits the requirements of our application. We considered factors such as the type of data to be processed, the model's performance metrics, and its compatibility with Flutter. Once the model was selected, we acquired a sufficient amount of labeled data and trained the model using popular Machine Learning libraries and frameworks like TensorFlow. Points noted during model training:
1. Training the model time: 5 hours.
2. Metrics considered **Mean Absolute Error**.
3. Loss per epochs reduced exponentially

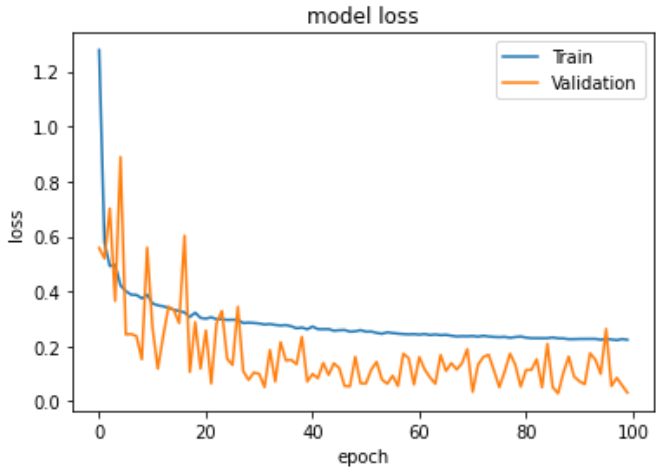


Fig.1 Model loss

Exporting the trained model

After training the Machine Learning model, we exported it in a format compatible with TensorFlow Lite, which is a lightweight version optimized for mobile devices. TensorFlow Lite provides an efficient runtime for executing Machine Learning models on Android devices. We followed the necessary steps to convert the trained model into the TensorFlow Lite format, ensuring compatibility with our Android application.

Integrating TensorFlow lite with Flutter

To incorporate the Machine Learning model into our Flutter-based Android application, we utilized the TensorFlow Lite Flutter package. This package provides the necessary APIs and utilities for loading and running TensorFlow Lite models within Flutter. We added the package as a dependency in our Flutter project and configured it to handle the model loading and inference processes.

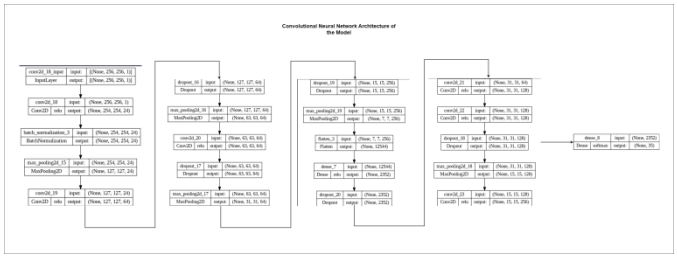
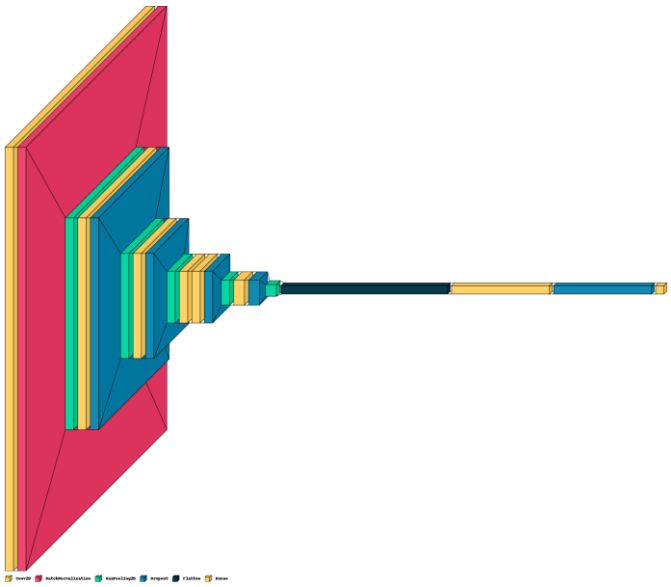


Fig. CNN model Per Layer Architecture

Details of the layers:

- Number of Convolution Layers: 6
- Batch normalization is used at the input layer. Implemented five MaxPooling2D layers for first to fifth layer.
- Activation functions used: Rectified Linear Unit (ReLU)and SoftMax
- Kernel Size for each layer is: 3 x 3

CNN Model Colored Architecture

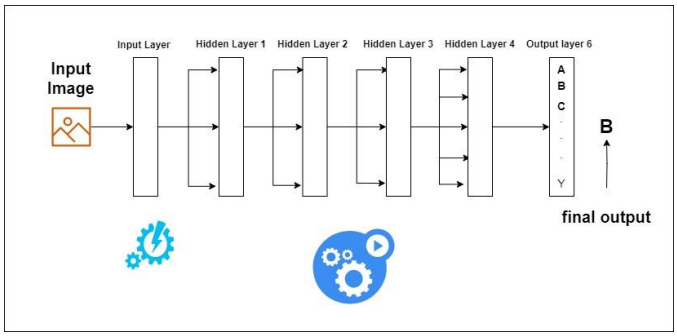


Details of the Colored Architecture of the model:

1. Yellow color represents Convolution 2D layers and Dense layers.
2. Red color represents Batch Normalization layers.
3. Green color represents Max Pooling 2D layers.
4. Blue color represents Dropout layers.
5. Dark Blue represents Flatten layers.

Workflow of the architecture

1. The image moves through the input layer where it is normalized using Batch Normalization method and then a Max Pooling layer enacts upon the image to extract the useful features is done. The data is passed to next layers.
2. The processing or the input layers is made using same Max Pooling and Dropout layers. Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass, and any weight updates are not applied to the neuron on the backward pass. The process goes on for the next four layers.
3. Then by flattening the input using the Flatten function to get output in a single dimensional array at the output layer



We conducted thorough testing to ensure the accuracy, reliability, and performance of the integrated Machine

Learning model within our Android application. We evaluated its behavior across different scenarios, edge cases, and real-world data inputs. Additionally, we optimized the model's inference process to achieve the best possible performance on mobile devices while considering factors like memory consumption and latency.

After testing the application on Computer and Android, we get the output for Random images like this

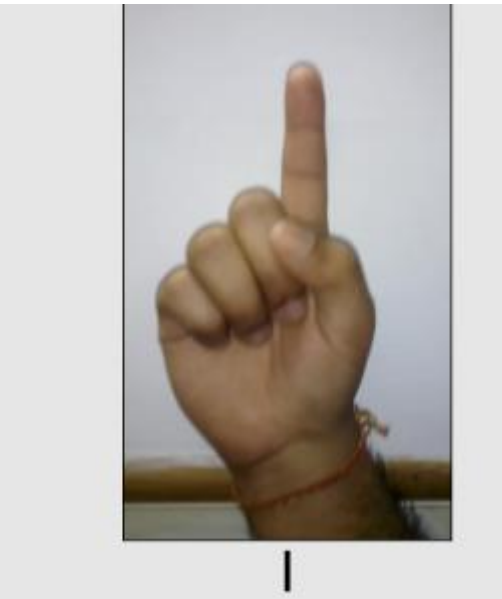


Fig. Sign for Alphabet I

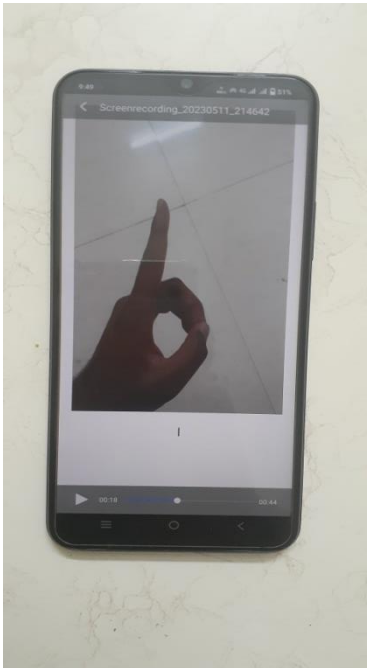


Fig. Output on android phone

Fig. Output for SIGN – O on Android Phone

Incorporating the Predictions into Applications

The predictions obtained from the Machine Learning model were seamlessly integrated into our Android application's user interface. We designed appropriate screens and widgets to present the predictions or analysis results to the users. The predictions were utilized to enhance the application's functionality, such as personalized recommendations, intelligent data analysis, or automated decision-making

7.CONCLUSION

The aim of this project is to predict the ISL alphanumeric hand-gestures in real time. The above work shows that it can be solved with better accuracy when we actually consider the segmented RGB hand-gestures. By applying depth-based segmentation we remove the overheads of dynamic background. The segmented RGB hand-gestures were fed to 3 layered CNN for training and testing in real time. We were able to achieve training accuracy of 89.30% and testing accuracy of 98.5%. Our model showed good accuracy while predicting results both offline and online.

REFERENCES

[1] B. L. Loeding, S. Sarkar, A. Parashar, and A. I. Karshmer, "Progress in automated computer recognition of sign language," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 3118, 2004

[2] A. Er-Rady, R. Faizi, R. O. H. Thami, and H. Housni, "Automatic sign language recognition: A survey," presented at the Proc. - 3rd Int. Conf. Adv. Technol. Signal Image Process. ATSIP 2017, vol. 9, pp. 1-7, 2017

[3] B. Bauer and K. Kraiss, "Towards an Automatic Sign Language Recognition System Using Subunits," presented at the International Gesture Workshop (pp. 64-75). Springer, Berlin, Heidelberg, 2002.

[4] J. Singha and A. Karen Das, "Recognition of Indian Sign Language in Live Video," 2013, arXivpreprint arXiv:1306.1301.

[5] R. Gross and V. Brajovic, "An Image Preprocessing Algorithm for Illumination Invariant Face Recognition," presented at the International Conference on Audio-and Video-Based Biometric Person Authentication Springer, Berlin, Heidelberg. vol.3, no. 5, 2018

[6] D. Kaur and Y. Kaur, "International Journal of Computer Science and Mobile Computing Various Image Segmentation Techniques: A Review," International Journal of Computer Science and Mobile Computing, 3(5), pp.809-814.

[7] "A Study of Image Segmentation Algorithms For Different Types Of Images," Int. J. Comput. Sci. Issues, vol. 7, no. 5, 2010.

[8] M. P. Paulraj, S. Yaacob, H. Desa, C. R. Hema, and W. M. R. Wan Ab Majid, "Extraction of head and hand gesture features for recognition of sign language," Int. Conf. Electron. Des. ICED, vol.2,no.5.January, 2008

[9] S. Joudaki, D. B. Mohamad, T. Saba,., A. Rehman., M. AIRodhaan, and A. Al-Dhelaan, "Vision-Based Sign Language Classification: A Directional Review VisionBased Sign Language Classification: A Directional Review," IETE Tech. Rev., vol. 31, no. 5, pp. 383-391, 2014

[10]G. Tofighi, SA. Monadjemi, and N. Ghasem-Aghaee, "Rapid Hand Posture Recognition Using Adaptive Histogram Template of Skin and Hand Edge Contour."2010 6th Iranian Conference on Machine Vision and Image Processing (pp. 1-5). IEEE.