International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 04 | April - 2023 Impact Factor: 8.176 ISSN: 2582-3930

Real Time Indian Sign Language Recognition

Shifa Mehreen, Chakrapani Anisetti, Chitikela Rohith Reddy, Rajapantula Sai Vinay

Mr Vuna Venkata Vidya Sagar, Dr. Pragnyaban Mishra.

Abstract: **Communication is essential for human survival and building relationships. India has a large deaf population of around 18 million people who use sign language to communicate within their community. To bridge the communication gap between the deaf and hearing population, machine learning technology has been used to create a model that can recognize Indian Sign Language (ISL).The model helps the deaf and hard of hearing community communicate with the hearing population by interacting through the camera and understanding each sign's meaning. Sign language is the community's own way of communication since there is no cure for deafness and hearing loss caused by various factors like aging, genetics, and environmental reasons. There are different types of sign languages. While ASL (American Sign Language) is the primary focus, our project aims to use ISL to help India's deaf and hard of hearing community bridge the communication gap with the hearing population.**

I. INTRODUCTION

Shifa Mehreen, Research Student, Dept. of CSE, GITAM(Deemed to be University) Visakhapatnam, mail id: 121910313005@gitam.in.

Chakrapani Anisetti, Research Student, Dept. of CSE, GITAM(Deemed to be University) Visakhapatnam, mail id: 121910313015@gitam.in.

Chitikela Rohith Reddy, Research Student, Dept. of CSE, GITAM(Deemed to be University) Visakhapatnam, mail id: 121910313033@gitam.in

Rajapantula Sai Vinay, Research Student, Dept. of CSE, GITAM(Deemed to be University) Visakhapatnam, mail id: 121910313041@gitam.in

Mr Vuna Venkata Vidya Sagar, Assistant Professor, Dept. of CSE, GITAM(Deemed to be University) Visakhapatnam, mail id: vvuna@gitam.edu

Dr. Pragnyaban Mishra, Professor, Dept. of CSE, GITAM(Deemed to be University) Visakhapatnam, mail id: pmishra4@gitam.edu

People who are deaf-dumb frequently communicate with one another through sign language. A sign language is nothing more than a collection of diverse gestures made by varied hand gestures, including movements, orientations, and facial expressions. Those who are deaf-dumb typically utilize these gestures to convey their thoughts. In public settings including banks, hospitals, and post offices, dumb-deaf people encounter communication barriers while communicating with normal people. The deaf occasionally need the assistance of a sign language interpreter to communicate their thoughts to hearing persons, and vice versa. By implementing such a system, the social divide between the hearing and hearing-impaired would be reduced.

The sign language used there will vary depending on the local culture and language. Indian deaf people communicate using Indian sign language (ISL). Speaking in English and using ISL is a common and well-developed method of communication for hearing-impaired people in India. For Indian Sign Language, distinct symbols are used for various alphabets. Both word-level gestures and finger spelling are used. This article describes a technique for automatically identifying static motions in the alphabet of Indian sign language. 17 letters from the English alphabet are among the symbols that are being examined for recognition.

The classification and recognition of the Indian sign language provided in real time by the dumb-deaf user are the key goals of the suggested approach. So, the algorithm's speed and simplicity are crucial. The system approach entails segmenting the hand based on skin color data, converting that segmented image to binary, and then extracting features from the binary image using distance transformation algorithms.

II. RELATED WORKS

This model deals with the “Real Time Indian Sign Language Recognition” for deaf and dumb people’s communication. Both manual and non-manual signals are used in sign language communication. Here we deal with manual signals. The goal of the sign language recognition system is to provide a quick and accurate way to translate text or voice, making it easier for hearing people to understand sign language.

Sign Language is basically hand gestures that require us to recognize what each sign means basing on already defined signs. These signs empower the deaf and dumb to visualize and understand what the other person wants to say. Since not everyone knows sign language, we tried to create a model that would help the ones that do not know sign language understand what a particular hand gesture means.

Machine Learning and Deep Learning is now being used everywhere. These powerful technologies have revolutionized the way businesses work and have made our daily lives simpler. By training our computers using the data available and allowing it to learn from that data has allowed us to predict outputs without having

to explicitly program. Image and Pattern recognition are few such techniques that can read the images displayed and guess what the image is.

We have created a model where a person is required to interact with the webcam and using Computer Vision and advanced algorithms like YOLO, the machine could recognize the hand gesture based on the data it was trained with and label the gesture accordingly. Artificial intelligence (AI)'s field of computer vision enables computers and systems to give out useful information from digital photos, videos, and other visual input. YOLO helps us to locate objects in particular images, localize and quickly recognize them based on the dataset given. It creates a bounding box around the object and labels it appropriately. We have firstly created a dataset for training. We built a custom dataset, then labeled the images using the Labeling module in Python. Then we did the preprocessing of images and feature extraction, where we extract the ROI, here the region of interest (ROI) is the hands, later image enhancement and segmentation were done. We then train the model using these images present in our custom dataset. We used YOLO as our object detection model, considering its speed and efficiency. We have various classes for various sings, which we give to the YOLO algorithm for training. Now while testing, when we interact with the webcam, it tracks our hand, creates a bounding box and predicts what gesture it is based on the dataset given to it while training.

As an output, we get a label and the accuracy, of what the sign is on the bounding box. Different signs have different accuracies. Accuracy depends on how well the model is able to classify the images based on the input data given. We have used various websites that help us understand how Indian sign language is different from the other sign languages. ISL uses both the hands for the English alphabets, hence we had to track both the left and right hands and also the finger positions, while creating a data set and while testing. There are research works on-going basing on ASL mainly, hence we tried using ISL to have a better understanding to how we can improve ISL reach higher levels in research. We used PyCharm to implement the code for creating a custom dataset for training purposes. We also used a Kaggle data as reference. We used python for the entire project. Python is a simple and easy to code language and has various predefined packages and has vast modules we can import from. The YOLO model was implemented in Google collab, where we cloned the YOLO's official repository from git and used it for training our model with the sign language data. It basically used transfer learning techniques for recognizing signs in real time based on what our dataset is. From the references, we were able to understand that there's still a lot of work that can be done in this field. Classifying images and converting it into grammatically correct text in real time, using NLP to get the right sentences, AI to automate the process and other modern techniques requires a lot of time, effort, better resources like large dataset and GPU systems. In this project we were only able to recognize what each sign means. We had different classes for different signs, through which we could predict what hand sign was being displayed using the video camera. Just like automated captions for a video like the YouTube videos, we would need to analyze the person's actions and gestures and give out a proper sentence as an output. We still have a long way to go.

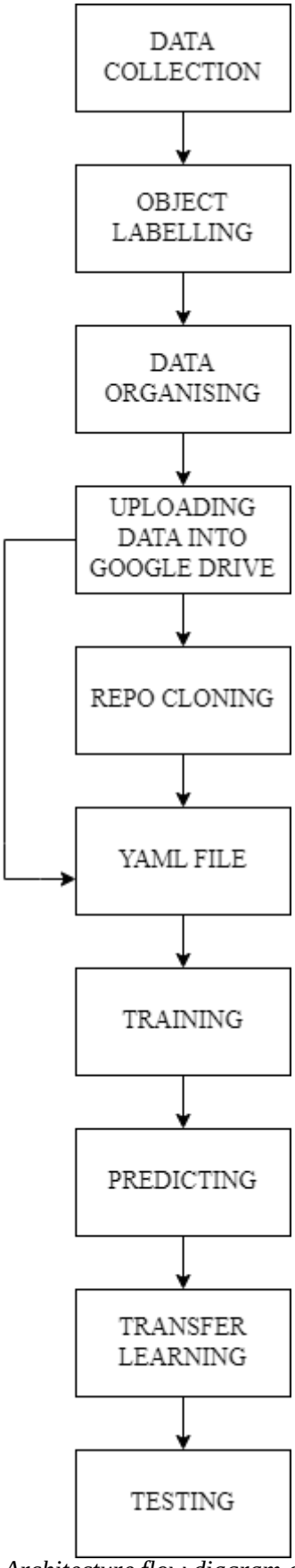


Fig: *Architecture flow diagram of the model*

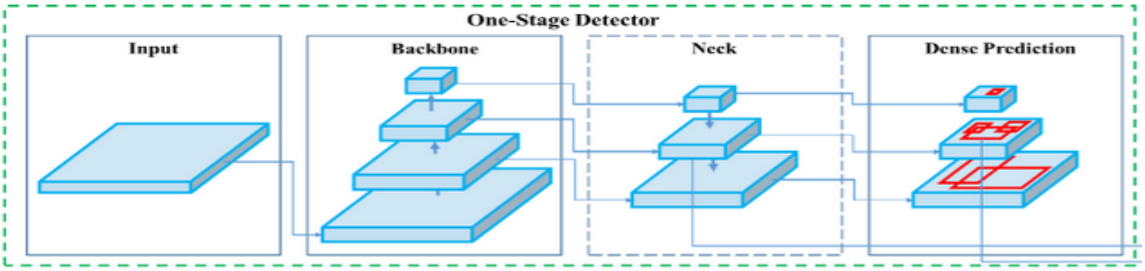


Fig: picture depicting One-stage detector

III. MATERIALS & METHODS

Object Detection is one of the computer vision tasks which uses image segmentation, image localization techniques. It is used to detect objects in an image or video stream. One of the earlier object detection modules is R-CNN, which uses a CNN with greater depth and Regions. As its successor, YOLO is far more advanced than R-CNN. YOLO is faster than R-CNN as it processes the whole image at once.

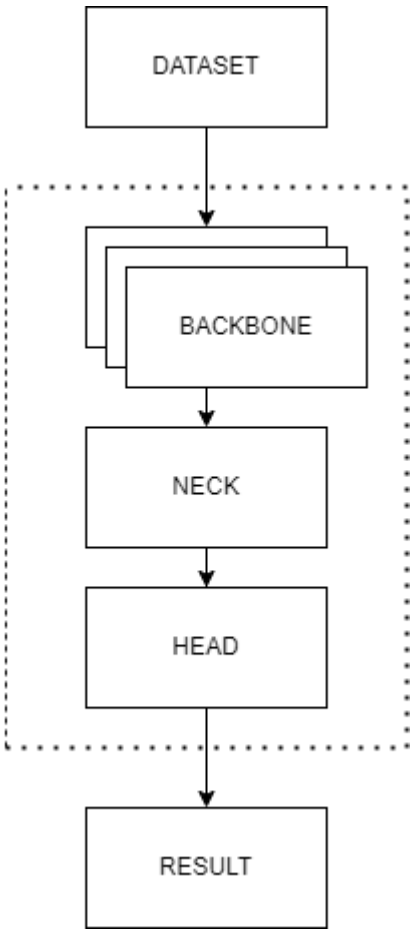


Fig: System flow of the model

Input layer takes the input from the dataset. Backbone Layers extracts the features from the input. The Neck Layer is similar to the bottleneck layer in GAN. it compresses all the extracted features. YOLOv5 has 3 layers which are used to predict the class probabilities and bounding boxes.

YOLOv5 is pre trained using the COCO dataset. COCO Dataset is a huge dataset which contains Common things which we come across in our daily life. It contains up to 90 classes, making it one of the best datasets for object detection. So in order to do transfer learning, we need to have a custom labeled dataset. If the dataset is not labeled, we must label them using annotation tools such labelImg, VGG image annotator.

We used labelImg as it provides multiple annotation formats like XML, TXT. labeling a object in image gives its respective object number and coordinates of the bounding box. data should be separated into train and validation folder and in each folder the images and their corresponding labels are to be separated into their respective folders. as YOLO model looks for these folders. The YAML file contains all the necessary data for our dataset to train. It contains the training data path and the validation data path. number of classes in the dataset and names of each object. This file should be uploaded to the data folder of the YOLO, because the yaml file of the previously trained dataset is present in the data folder.

Once the training is completed, the results are stored in the runs folder. It contains the weights which are obtained after training. These weights are then used for testing, and it also predicts a few classes to test the accuracy of the trained model, if we are not satisfied with the accuracy then the data can be trained once again by increasing the epochs. With the updated weights assigned to the data, we can test the model by giving data from external sources which can be many formats, –0 is used to access webcam and .mp4 is used for video.

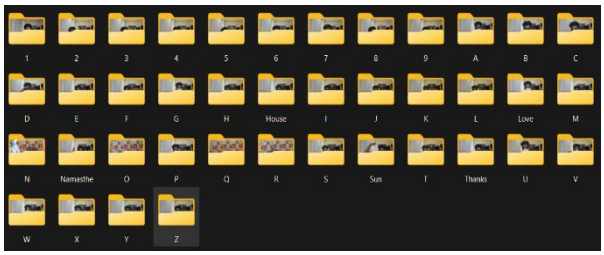
IV. IMPLEMENTATION

1) Data is collected using a python program which uses OpenCV module.

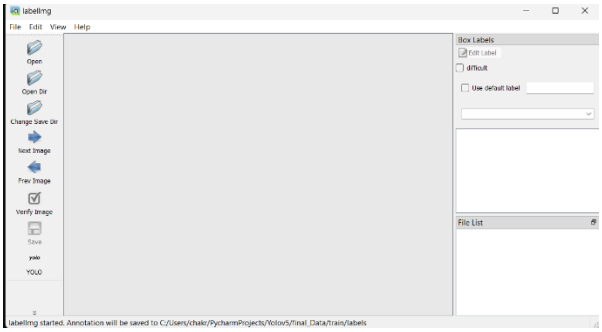
```
import os
import cv2
import time
import uuid

IMAGE_PATH = 'Data'

labels=['1','2','3','4','5','6','7','8','9']

no_of_images = 9

for label in labels:
    img_path = os.path.join(IMAGE_PATH, label)
    os.makedirs(img_path)
    cap = cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
    time.sleep(5)
    for imgnum in range(no_of_images):
        ret,frame = cap.read()
        imagename = os.path.join(IMAGE_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imagename,frame)
        cv2.imshow('frame',frame)
        time.sleep(2)
        if cv2.waitKey(1) & 0xFF==ord('q'):
            break
    cap.release()
```

- os module is used to assign paths to the folders which store the images.
- the opencv (cv2) module contains many functions, here we used VideoCapture() function, argument – 0 is used to access the webcam and release() is used to release all the resources.
- the time module is to add delay.
- the uuid module is to name the captured images.

2) DATA folder contains all the images that are collected



3) Once all the data was collected, we started labeling, we used LabelImg which is an annotation tool.



- open dir is used to open the directory which contains the images.
- change save dir is used to open a path which is used to save the annotated files.
- annotated file should be changed to yolo as we are working with yolo
- In Colab notebook, we start with mounting our notebook with google drive.

```
from google.colab import drive
drive.mount('/content/drive')
```

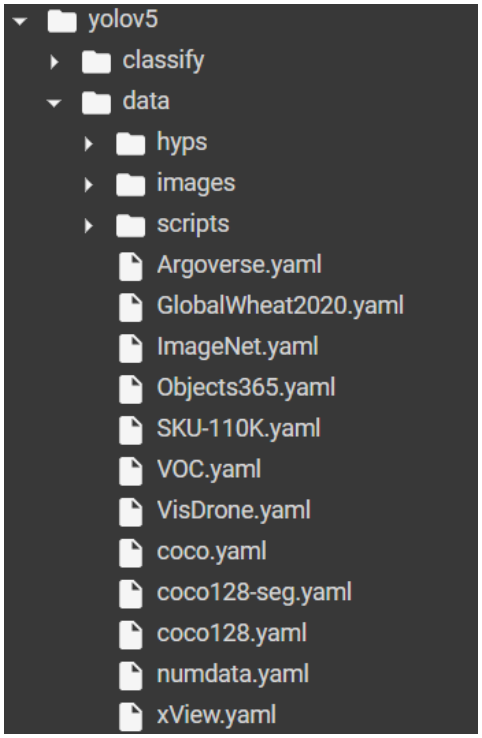
- We cloned the official YOLOv5 repository into our notebook using the following command.

```
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
%pip install -qr requirements.txt
```

- Now the yolov5 folder is created in our notebook environment which contains all the source code.
- A YAML file is created which contains the training and validation paths, number of classes and list of classes.

```
train: /content/drive/MyDrive/numData/train/images
val: /content/drive/MyDrive/numData/val/images

nc: 9

names: ['1','2','3','4','5','6','7','8','9']
```

- This file is then loaded into the data folder which is present in the yolov5 folder.



- Now we will train the model with our custom dataset using the following command

```
!python train.py --img 640 --batch 2 --epochs 10 --data numdata.yaml --weights yolov5s.pt --nosave --cache
```

- train.py is one of the YOLOv5 source code.
- arguments : img – image size, batch – batch size, epochs – number of iterations.
- data – YAML file, weights – weights of pre trained model.

```
Overriding model.yaml nc=80 with nc=9
```

- nc is overridden by our numdata.yaml.

```
Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
  7/9     0.545G    0.02095   0.008398    0.01413          1        640: 100% 563/563 [01:00<00:00,  9.31it/s]
             Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 57/57 [00:03<00:00, 16.30it/s]
               all        225        225      0.977          1      0.995      0.761

Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
  8/9     0.545G     0.0178   0.007587    0.01119          3        640: 100% 563/563 [01:00<00:00,  9.24it/s]
             Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 57/57 [00:02<00:00, 19.82it/s]
               all        225        225      0.983          1      0.995      0.789

Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
  9/9     0.545G    0.01718   0.007856    0.01116          0        640: 100% 563/563 [01:01<00:00,  9.15it/s]
             Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 57/57 [00:02<00:00, 19.30it/s]
               all        225        225      0.989      0.998      0.995      0.825
Model summary: 157 layers, 7034398 parameters, 0 gradients, 15.8 GFLOPs
             Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 57/57 [00:04<00:00, 11.55it/s]
               all        225        225      0.989      0.998      0.995      0.825
                 1        225         25      0.989          1      0.995      0.881
                 2        225         25      0.985          1      0.995      0.683
                 3        225         25          1      0.982      0.995        0.8
                 4        225         25      0.994          1      0.995      0.791
                 5        225         25      0.982          1      0.995      0.863
                 6        225         25      0.987          1      0.995      0.843
                 7        225         25      0.989          1      0.995      0.849
                 8        225         25      0.985          1      0.995      0.839
                 9        225         25      0.989          1      0.995      0.877
Results saved to runs/train/exp
```

- We can see the trained model is saved in the runs folder.
- We can see the model has predicted some classes.
- This is done using the ML technique called Transfer Learning.
- Transfer Learning is the process by which we replicate the results of a pre-trained model with a custom model.
- It saves a lot of time.

- Now we test our input using the following command:

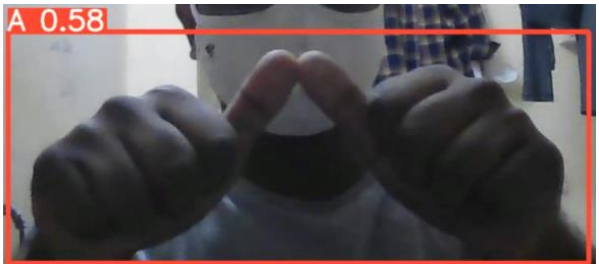
```
!python detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt
--img 640 --conf 0.35 --source /content/WIN_20230319_19_49_54_Pro.mp4
```

- arguments: weights – weights which are obtained by training our custom dataset, img – image size, conf – confidence score, source – input

```
video 1/1 (151/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 (no detections), 11.5ms
video 1/1 (152/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 1 3, 11.1ms
video 1/1 (153/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 (no detections), 10.4ms
video 1/1 (154/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 1 3, 9.9ms
video 1/1 (155/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 1 3, 9.9ms
video 1/1 (156/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 (no detections), 9.7ms
video 1/1 (157/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 1 3, 10.1ms
video 1/1 (158/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 (no detections), 9.8ms
video 1/1 (159/428) /content/WIN_20230319_19_49_54_Pro.mp4: 384x640 (no detections), 10.1ms
```

- We can see that the video input is getting converted into frames and it detects labeled objects within those frames.
- We can update YAML files to obtain pre-trained models for words and alphabets.

V. RESULTS



V. Conclusion & Future

In this way we created a model which can be used to interact with deaf and dumb people more effectively. With a huge, labeled dataset we can train the model with classes present in it, which reduces training time and with a higher GPU system we can even label the videos by converting them into frames and each frame is labeled with a class. It solves the problem of complex signs.

We could later automate the process using AI, where the system can learn new signs and predict outputs on its own. Using a better camera, we could track the movements of hands and body, including the elbows and facial expressions, to understand different signs.

We were now only able to recognize a few alphabets and numbers that were displayed using hand gestures. We aim to recognize continuous hand, body, and facial movements, to form sentences and fully understand what the person is trying to express and in what tone. Further, one could interact with the video camera using sign language and the other could read the text developed from that sign language, i.e., sign to text translation. This could allow hearing- and hearing-impaired people to interact in the video calls easily. People could learn sign language in an easier manner. This will help the differently abled people express their thoughts, ideas and opinions reach a larger audience and feel empowered overall.

References

SOURCE:

[1] Indian Sign language - ISL

[2] Talking Hands - Talking hands

[3] Indian Sign language and Training Centre

TECHNOLOGIES:

[3] YOLOv5 - YOLOv5

[4] LabelImg - LabelImg

REFERENCE DATASET:

[5] Indian Sign Language

JOURNALS:

[6] Automatic Indian Sign Language Recognition System

[7] Real time Indian Sign Language Recognition System to aid deaf-dumb people

[8] Sign language recognition using image based hand gesture recognition techniques