VOXVISOR

V. Avinash[1], K. Dinesh Charan Raj[2], Ch. Naveena[3], A. Madhavika Santhoshi[4], A. Jai Vardhan[5]
[1,2,3,4,5] Department of Artificial Intelligence and Machine Learning, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh, India

***Abstract*—The communication barrier between hearing and non-hearing individuals, particularly those with speech impairments, remains a significant challenge. Traditional sign language interpretation methods, often reliant on manual techniques, are time-consuming and limited in their ability to adapt to diverse signing styles. To address these limitations, we propose a deep learning-based system, Voxvisor, for real-time sign language recognition and translation into audible speech. Voxvisor incorporates advanced computer vision techniques, including key-point detection, optical flow, and YOLO (You Only Look Once) feature extraction, to accurately identify and classify sign language gestures. By leveraging deep learning architectures such as CNNs, RNNs, and LSTMs, Voxvisor can effectively learn from a comprehensive dataset of sign language videos, capturing both spatial and temporal characteristics of gestures. Compared to existing manual methods, our approach offers several advantages: real-time recognition, adaptability to various signing styles, improved accuracy. By bridging the communication gap between hearing and non-hearing individuals, Voxvisor has the potential to significantly improve the quality of life for those with speech impairments and promote social inclusion.**

***Index Terms*—Deep learning, YOLOv5 (You Only Look Once), Optical flow, Real-time recognition, Sign gesture interpreter.**

I. INTRODUCTION

Sign language plays a crucial role in communication for the deaf and hard-of-hearing community. However, there are often significant barriers to effective communication between deaf and hearing individuals. It is essential to bridge this gap to promote inclusivity and accessibility. This project focuses on creating a reliable sign language gesture recognition system utilizes the YOLOv5 object detection framework developed by Ultralytics [1] as the foundation for our sign language gesture recognition system. While YOLOv5 provides a robust and efficient base model, this study focuses on adapting the model for sign language data, optimizing hyperparameters, evaluating performance on a new dataset.

YOLOv5 has shown outstanding performance in real-time object detection tasks, making it an excellent choice for this purpose. By utilizing the advantages of YOLOv5, including its high accuracy and speed, we aim to develop a system capable of accurately and efficiently recognizing a wide range of sign language gestures.

This research will aid in the advancement of assistive technologies for the deaf community. If successfully implemented, this system could enhance communication in various environments, such as education, healthcare, and social interactions.

This paper follows as mentioned structure to go through the modern approach in finding the solution to bridge the communication barrier between the hearing and non-hearing people. Section II mentions the background study, research and survey of different literatures. Section III provides the materials and methodologies used to collect dataset and preprocessing them to attain the high performance of the Yolov5 model. Section IV offers the case study of the results and the performance of the YOLOv5 model. Section VI presents the conclusion and future scope of Indian Sign Language.

II. BACKGROUND STUDY

Indian Sign Language recognition can be performed using data from various sources, such as videos, photographs, wearable sensors, and more. Numerous research studies have explored hand gesture recognition. Early approaches to this field often involved the use of hand gloves equipped with cables, sensors, LED markers, or other devices [3]. These early methods worked well only when the lighting was consistent. However, accurately recognizing hand gestures can be quite tricky. Researchers have explored many different features, such as skin color and hand movement speed, to help identify hand motions [4]. In spite of these traditional techniques many authors introduced SLR [22] using Machine Learning [5], [6]. There are numerous different techniques and methodologies that belong to this group, some of the well-known methods include naïve Bayes, random forest, K-nearest neighbor, logistic regression and the support vector machine [5], [6]. All these may include in the training phase, those can be supervised or unsupervised.

Those basic machine learning approaches have become older as the new approaches introduced in the recent times, which are nothing but deep learning methods and models. Those models include most famous and powerful algorithms such as recurrent neural network (RNNs), convolution neural network (CNN). In [7] the authors trained CNN in addition they optimized the performance of the CNN by fine-tunning of the model by changing some of the relevant hyper-parameters that define training process [7].

In [8] the researchers started with a pre-trained Inception V3 model, originally developed by Google and trained on the massive ImageNet dataset. To adapt this model for their specific task of recognizing gestures in video frames, they fine-tuned it using the ImageNet weights as a starting point. In [9] the authors significant research effort was undertaken to categorize Indian Sign Language gestures, resulting in a dataset of 140 classes encompassing finger-spelled numbers, English alphabets, and common phrases. A Kinect sensor was utilized to capture the dataset, acquiring 640x480 RGB images along with their corresponding depth data. Depth values were recorded for each pixel, establishing a direct mapping between image pixels and their depth information. Given that subjects stood with their hands extended, the hand in the image exhibited the least depth. This was validated by applying a pixel mask to filter out depth values exceeding a specific threshold. However, this masking process inadvertently excluded the palm region, which displayed significant inconsistencies. Consequently, this portion of the dataset was not used for training the Convolutional Neural Network (CNN). To address this, unsupervised learning was employed, specifically the K-means clustering algorithm. SIFT feature mapping and Gaussian masks were utilized to extract relevant features and train the dataset. The final accuracy achieved exceeded 90%.

Nandy et al. [10] classifies the gestures by splitting the data into segmented features and employs Euclidean distance and K-Nearest Neighbors. Similar work by Kumud et al. [11] shows how to do continuous recognition. Their research proposes extraction of still frames from videos, pre-processing data, extracting data frames and other features. Pre-processing is done by converting the video into RGB images or frames with same dimension. Skin color segmentation with the HSV was used to extract skin region and were converted to binary form. Gradient calculation between the frames is done to extract the key frames and oriental histogram is used for features extraction.

Huang et al. [12] developed a sign language recognition system using a Kinect sensor and 3D convolutional neural networks. They utilized 3D CNNs to capture both spatial and temporal features directly from the raw data, which aids in extracting meaningful features to handle the significant variations in hand gestures. This model's effectiveness was validated on a real-world dataset comprising 25 signs, achieving an impressive recognition rate of 94.2%. Huang et al. [13] proposed a sign language recognition system using the RealSense depth camera. They collected a total of 65,000 image frames representing 26 alphabet signs, dividing the data into a training set of 52,000 frames and a testing set of 13,000 frames. The deep neural network model was trained and classified using a deep belief network, achieving an impressive accuracy of 98.9% with the RealSense camera and 97.8% with the Kinect sensor.

Pigou et al. [14] contributed their efforts to a sign language recognition system using Microsoft Kinect and a CNN-based approach. In this system, they employed thresholding, background removal, and median filtering as preprocessing steps. They implemented the Nesterov Accelerated Gradient (NAG) optimizer, achieving a validation accuracy of 91.7% for recognizing Italian gestures. Yang and Zhu [15] created a system that uses videos to understand Chinese Sign Language (CSL). This system relies on a type of artificial intelligence called Convolutional Neural Networks (CNNs). They gathered data using 40 commonly used words in CSL. Their approach makes it easier to identify the hands in the videos and avoids losing important information when analyzing the signs. They tested two different training methods

for the AI, called Adagrad and Adadelta, and found that Adadelta worked better. Bheda and Radpour [12] created a system that recognizes American Sign Language (ASL) letters and numbers. Their system uses a type of artificial intelligence called Convolutional Neural Networks (CNNs). The CNN has a specific structure with multiple layers for processing information. The images used to train the system were cleaned up using a technique called background subtraction. This system achieved an accuracy of 82.5% for recognizing letters and 97% for recognizing numbers.
The authors of [17] had implemented their custom convolutional neural networks, for recognizing the sing from a video frame. The MNIST (Modified National Institute of Standards and Technology database) [17]. The MNIST is one of the well-known datasets, although the system performed well it still struggles in real-time scenarios with uneven or noisy backgrounds. Sign Language Recognition System using Machine Learning [23], the authors made a step forward by not to just rely on the simple hand gestures also focused on the facial expressions and key-point detection. Although their idea was good to see but it is computationally expensive while compared to other approaches.
Melek Alaftekin and other authors of [25] utilized Yolov4-CSP algorithm by adding few add-ons such as Mish activation function, complete interaction of union (CioU) loss function and transformer block. Their system obtained 98.95% precision, 98.15% recall, 98.55% F1 score and 99.49% mAP results in 9.8ms. This proposed system can able to detect numbers in Turkish sign language.

III. MATERIALS AND METHODS

The techniques and resources that were used in this study to achieve the hand gesture recognition that this paper focused on are assigned to this part. Fig. 1 resembles the flow chart of the methodology for sign language recognition.
The proposed method was divided into different steps. Firstly, the hand images were collected from the various resources from the internet for Indian sign -language hand gesture recognition and underwent data augmentation to create a hand motions dataset. The captured image is passed through an annotation format to draw a Bounding box-based hand detector to extract hand regions. Bounding boxes were manually drawn around specific objects in the images to annotate them. Once the hand has recognized and transferred to a better Yolo (You Only Look Once) deep learning model, then the hand region is extracted. This model was then optimized and trained on the created datasets. The dataset has 35 different gesture classes, such as Alphabets and decimal numbers. To verify the detection performance, evaluation metrics were produced. The best model was chosen for the best hand detection across many images.

A. Dataset Collection

Data 1: This dataset [data set reference] contains about 3000 images annotated with 35 different classes in different scenarios to make the model to adapt to every situation as possible.
Data 2: This is a custom dataset which is annotated with 35 classes i.e. A- Z, 26 English alphabets and 1-9, 9 decimal numbers. The data set is annotated manually by drawing a box that locates the gesture.

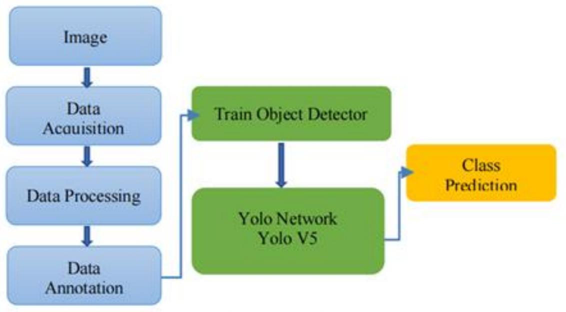

Fig. 1. The flowchart of the proposed hand gesture recognition.

Then we labelled the dataset in roboflow [roboflow reference] tool present in the internet.

B. Data Acquisition

In this paper, the ISL images were collected from the image database for tiny hand gesture recognition. This dataset [2] has been collected from the open-source tool available in the internet. There are total of 35 classes with about 2000 images. Our classes started from 1 to 9 and A to Z, which were finger-pointing different positions. The Indian Sign gestures in our dataset are shown in Fig. 2.

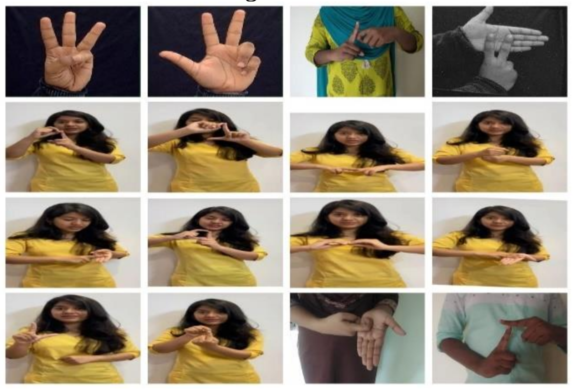

Fig. 2. Sample data of each class.

C. Data Pre-processing

By artificially increasing the dataset, data augmentation is a key strategy for creating variations of the training and testing datasets. This step consists to utilize the augmentation techniques such as brightness transformation, randomly altering rotation, motion blur, blurring, and the scale of an input image necessitates that a model contemplates what an image subject looks like in a diversity of positions. Each image was repeated for reading and training, both for the left and right hand, by flipping it horizontally, and sometimes capturing the respective image of those hands to make the set more accurate, using a YOLO setup with a total of images from the dataset. Additionally, each image for the testing set was captured and labelled. Before moving on to post-processing, it is vital to perform data pre-treatment so that we can determine the type of data we have collected and which portions will be relevant for training, testing, and improving accuracy. This part presents the system or methods used to classify, select, and process as well as analyse data and its recognition of characters is discussed. The following methodology is employed to collect data in the form of images, preprocess the data, and then feed the processed data to our model.

1. *Annotating manually:* The procedure of annotation, the training and validation set images were originally 240×240 pixels in size. We utilized the internet tool Roboflow to construct the bounding boxes for each image (www.roboflow.com). This page facilitates making data labels and annotating in the desired format. The images were annotated using the Roboflow Annotate, which is a self-serve annotation tool, and that greatly accelerates the transition from untrained and deployed computer vision models to raw images. After manual drawing and categorization of bounding boxes, this tool made it possible to change just one annotation entire the whole dataset.

2. *Object detection:* The object detection model is trained in this section. We concentrate on the most recent deep learning-based object detection models, albeit any detector can be used. In the following part, we'll go into more detail about our training methods. To determine the existence, quantity, and placement of objects in a picture, object detection models are used. Drawing a box for each object of interest on each image was necessary for the image annotation, which enables us to determine the precise location and quantity of objects in an image. In contrast to image classification, where the class placement within the image is irrelevant because the entire image is designated as one class, the class location is a parameter in addition to the class. Bounding boxes and polygons are examples of labels that can be used to annotate objects inside a picture. Find the existence of things in an image using a bounding box and the types or classes of the objects you find.

A) Input: An image that includes one or more items, like a photo.

b) Output: One or more bounding boxes with class labels.

3. *Image data labelling with bounding box:* We have also produced a dataset with bounding box so that we may utilize the characteristics of the deep learning detection technique. We randomly choose a few images from each class in the dataset and choose to label the bounding boxes. The most popular annotation shape in computer vision is the bounding box. Angular boxes called bounding boxes are used to specify where an object is located inside an image. Both two-dimensional (2D) and three-dimensional (3D) models are possible (3D). Polygons or rectangular shapes were manually drawn to annotate the object's edges and to mark each of the object's vertices. The x_center, y_center, width, and height of an object's boundary show its exact location in that image. As shown in Fig. 3, the rectangular shapes are used to label different hands.

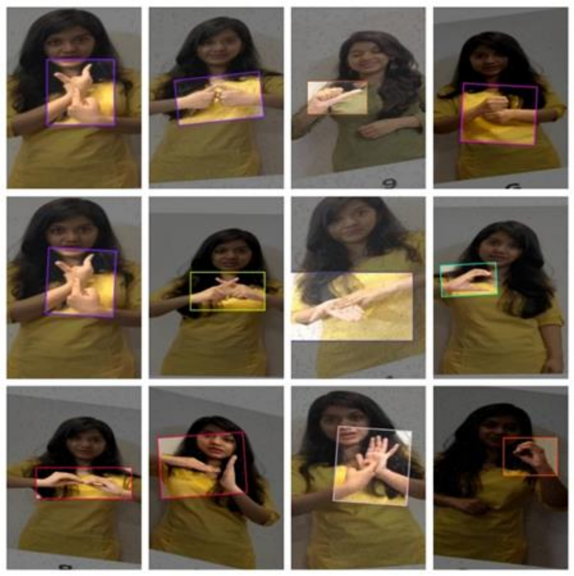

Fig. 3. Labelling different hand classes with bounding boxes.

D. Labelled Dataset

In a labelled dataset, each element of the unlabelled data is given a meaningful "label", "tag" or "class" that makes it more desired or instructive to identify it. Bounding box inference in the training detection model continues until all unlabelled images have been manually fully tagged. In our model we annotated the dataset, we introduce seven different gesture classes, such as English Alphabets and decimal numbers.

E. Structure of YOLO Algorithm

1. *You Only Look Once (YOLO):* YOLO means You Only Look Once is a method that detects all objects in a frame or image in a single shot. Mainly, YOLO employs a single, fully convolutional network (FCN) comprised entirely of convolutional layers to identify the objects present in an image. The YOLO approach segments an image into a grid of cells, where each cell is responsible for object localization, determining the number of bounding boxes, and computing class probabilities. The dataset is collected from various people with various complex backgrounds at different positions, such as variable illumination, gesture variations, and low resolution. Labelling images is essential for good computer vision models. All the images are annotated and labelled manually with Roboflow Annotate which represents a self-serve annotation tool. In this study, we provide a dataset called "Final_ISL", to which we add bounding boxes to roughly 3000 images in order to make use of the potential of object detection techniques. After the first step of preprocessing and the manual annotation, the second one is training the deep learning models using modern YOLO algorithms YOLO v5. To understand the algorithms which we are proposing, the diagram presented in Fig. 1 shows the detection of objects. At first, the first step in the training process is to gather the data, and the second is to label it. Our dataset is annotated using the YOLO format, providing specific values that are subsequently used during the model's training. We feed the dataset to the YOLO v5 model afterwards, after it has been annotated with YOLO annotation. There are now a variable number of images in our dataset.

2. *YOLO v5 model:* The Backbone, Neck, and Head architectural components of the YOLOv5 network are shown in Fig. 4.

YOLOv5 Backbone: CSPDarknet, is employed to extract image features, incorporating cross-stage partial networks.

YOLOv5 Neck: It makes use of PANet to create a feature pyramid network that is then passed to the Head for prediction after the features have been aggregated.

YOLOv5 Head: Its layers produce predictions for object detection from the anchor boxes.

YOLOv5 is quick and lightweight, and it uses less computing power than other current state-of-the-art architecture models while maintaining accuracy levels that are comparable to those of current state-of-the-art detection models. It is significantly faster than other YOLO versions. YOLOv5 leverages CSPNET as the basis for extracting feature maps from images. In order to improve information flow, it also makes use of the Path Aggregation Network. For the following reasons, we have chosen YOLOv5 because it incorporates advantageous features such as an advanced activation function, a user-friendly guide, hyperparameter tuning, and data augmentation capabilities. It can be trained computationally quickly with minimal resources, thanks to its lightweight architecture. The size model can be utilized with mobile devices because it is relatively tiny and light.

YOLOv5 presents several key differences compared to previous versions in the YOLO series:
1. Multiscale: utilize FPN to improve the feature extraction network rather than PAN, which will make the model easier to use and more quickly.
2. Target overlap: identify nearby positions using the rounding method such that the target is mapped to several central grid points all around it. Yolov5 is a continuation of the YOLO series' most recent iterations. It is more manageable and, in general, cozier to utilize throughout training. Its architecture may be modified with equal ease, and it can be exported to numerous deployment environments.

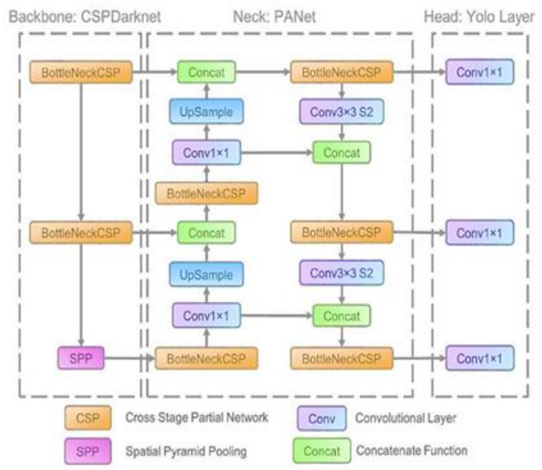

Fig. 4. The general architecture of the YOLOv5 network

YOLO models have several algorithmic parameters, and understanding their impact is critical for optimizing the model's performance for specific tasks. Below are some key parameters in YOLO models and their effects:

Input Size: This determines the resolution of the input image. While larger input sizes can improve model accuracy, they also increase computational cost.

Anchor Boxes: These are predefined boxes of various shapes and sizes used for predicting object locations and dimensions. The number and aspect ratio of anchor boxes play a significant role in the model's accuracy.

Batch Size: Training speed can be increased with larger batch size however; this comes at the cost of higher memory requirements.

Confidence Threshold: This parameter filters out predictions with low confidence. Raising the threshold

reduces false positives but may also increase false negatives.
NMS Threshold: Non-Maximum Suppression (NMS) removes overlapping bounding boxes. The NMS threshold sets the permissible overlap level between boxes. A higher threshold removes more overlaps but might also discard some true positives.
Backbone Architecture: The backbone architecture extracts feature from the input image. Different architectures vary in complexity and influence both the accuracy and speed of the model.
CSP: Cross Stage Partial Network
SPP: Spatial Pyramid Pooling
Conv: Convolutional Layer
Concat: Concatenate Function
For example:
Conv1x1, Conv3x3 S2, BottleNeckCSP layers are integral to feature extraction.
Neck (PANet), Head (YOLO Layer), and Backbone (CSPDarknet) components collectively enhance detection efficiency.
Training Parameters: Parameters such as learning rate, weight decay, and optimizer have a significant impact on the training process and the model's overall performance.
The parameters of YOLO models directly influence their accuracy, speed, and memory requirements. Selecting the most suitable parameters for a specific task demands experimentation and fine-tuning to achieve optimal results.

IV. EXPERIMENTS AND RESULTS

1. *Evaluation Metrics*

In this section, we discuss the experiments performed using Yolov5 algorithm. We implemented and test the model during our experiments to train it for our custom dataset which is different from publicly available datasets. The evaluation metrics are described after completing the model training and the model testing. To evaluate the performance of the proposed hand gesture recognition model, several metrics were employed, focusing on recognition accuracy, detection capabilities, and computational efficiency. Among these, average precision (AP) was used to assess performance. AP is calculated as the area under the precision-recall curve across different detection thresholds. Eq. (1) contains a definition of the Average Precision (AP) equation.[26]

$$AP = \int_{1}^{0} P_r(R_C)\, dR_C \qquad (1)$$

To assess the model's accuracy and efficiency, we calculated precision, recall, and F1-score. Accuracy is determined by comparing predicted bounding boxes to ground truth boxes. Additionally, we employed Equations (2), (3), and (4) to derive precision, recall, F1-score, and accuracy using True Positives (TP), False Positives (FP), and False Negatives (FN). Precision (Pr), as defined in Equation (2), represents the ratio of TP to all expected positives (TP+FP). Consequently, it is a critical metric for evaluating the cost associated with FP instances.[26]

$$P_r = \frac{T_p}{T_p + F_p} \qquad (2)$$

If the predicted bounding box doesn't overlap with the actual hand region (ground truth), it's classified as a False Positive (FP). Conversely, if the prediction correctly identifies the hand's location, it's a True Positive (TP). Recall measures how well the model detects all the actual hands in the video. It's calculated as the ratio of correctly detected hands (TP) to the total number of actual hands present (TP + FN), and is also known as sensitivity. A False Negative (FN) occurs when the model fails to detect a hand that Is actually present in the video frame.[26]

$$R_c = \frac{T_p}{T_p * F_N} \qquad (3)$$

The F1-score gives us a good overall idea of how well the model performs, taking into account both how accurately it identifies things (precision) and how many of the actual things it finds (recall). As shown in Equation (9), the F1-score considers both of these aspects. It's especially useful when we need a good balance between identifying things correctly and making sure we find most of them. A perfect F1-score of 1 means the model is doing both perfectly.[26]

$$R_c = \frac{2 * P_r * R_c}{P_r * R_c} \qquad (4)$$

Mean Average Precision (mAP), a popular metric for evaluating object detection models, is calculated by averaging the AP values for all the different object classes. This gives us a single, overall score that tells us how well the model performs across all the objects it's trained to detect. The Eq. (5) gives the Mean Average Precision (mAP).[26]

$$mAP = \frac{\sum_{q=1}^{0} AveP(q)}{Q} \qquad (5)$$

Where Q is the number of queries in the set, q is the query for average precision. The mAP is the mean value of average precision for the detection of all classes and is an indicator generally utilized to estimate how good a model is. The FPS identifies how many images can be correctly identified in a single second. GPU utilization refers to the use of GPU RAM when evaluating various detection strategies.[26]

2. *Results of YOLOv5 Model*

The output of the various classes of Indian Sign Language Detection is shown in Fig. 5. The bounding box aimed to encompass as much of the hand as possible. This is especially important when dealing with large objects, as it helps the model accurately identify the specific sign being performed. Essentially, the model zooms in on the object and then determines the most likely class based on its characteristics. To evaluate the effectiveness of different Indian Sign language detection methods, we performed several Experiments.

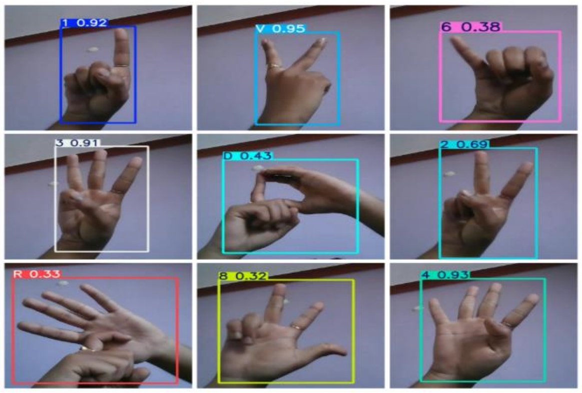

Fig. 6, 7, 8, 9 and 10 shows the Confusion metrics, F1-Score / F1-Curv, Precision-Confidence Curve, Recall-Confidence Curve and mAP at 0.5% respectively. Which shows the outperformance of the YOLOv5 model in object detection.

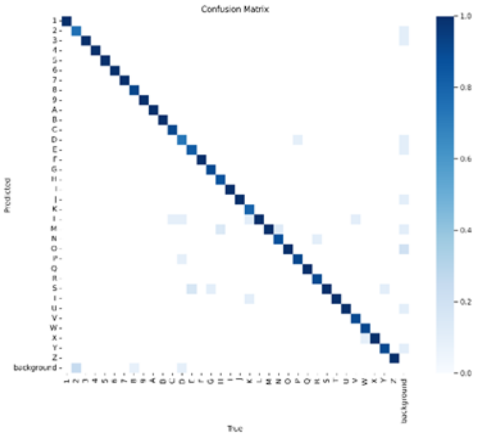

Fig. 6. Confusion metrics

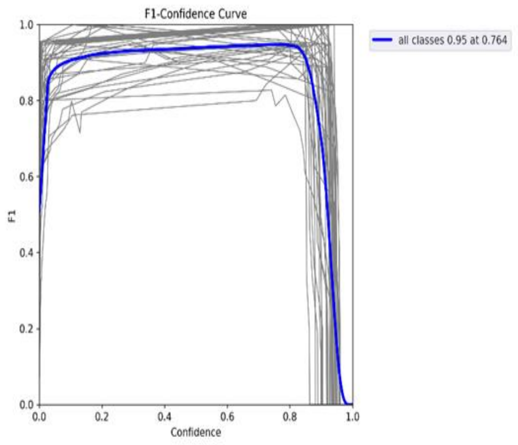

Fig. 7. F1-Score / F1-Curve

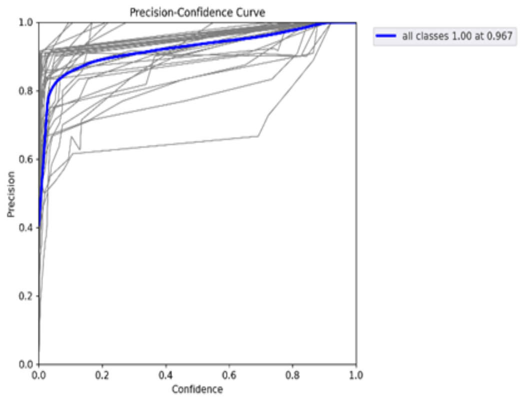

Fig. 8. Precision-Confidence Curve

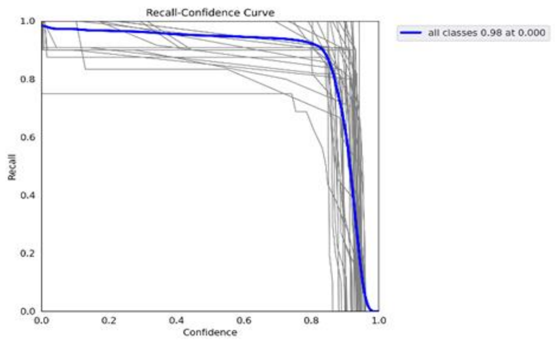

Fig. 9. Recall-Confidence Curve

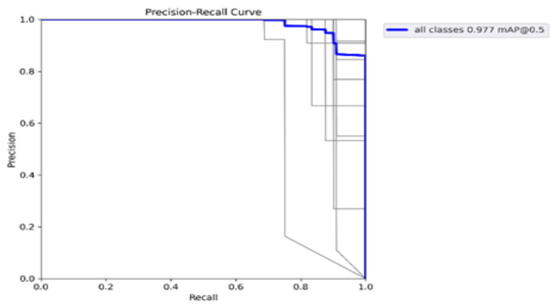

Fig. 10. Precision-Recall Curve

V. CONCLUSION AND FUTURE SCOPE

This research paper successfully demonstrated the feasibility of utilizing the YOLOv5 object detection model for recognizing Indian Sign Language (ISL) gestures. The model was trained on a dataset of 3000+ images encompassing 1 to 9 decimal numbers and 26 alphabets which in total 35 classes and achieved an accuracy of 97.7% on the test set. This accuracy level indicates the model's potential for real-world applications, such as assistive communication devices for the deaf community. The YOLOv5 architecture, with its efficient design and high detection speed, proved to be suitable for this task. The model was able to effectively detect and classify various ISL gestures with a high degree of accuracy, demonstrating its capability to handle the complexities and nuances of human hand movements.

This project serves as a foundation for further research and development in the field of ISL recognition. Some potential future directions include:

1) Increase the size and diversity of the training dataset to improve model robustness and generalization.
2) Include more complex gestures, dynamic sequences, and real-world scenarios with varying lighting and backgrounds.
3) Implement a system for continuous learning to adapt the model to new gestures and variations in individual signing styles.
4) Develop a user-friendly interface for interacting with the system, such as a mobile application or a wearable device.
5) Optimize the model for real-time inference on edge devices like mobile phones and embedded systems.

REFERENCES

[1] Comprehensive Guide to Ultralytics YOLOv5

[2] https://docs.ultralytics.com/yolov5/

[3] Custom Dataset Annotated manually using roboflow an open-source tool available on internet. https://app.roboflow.com/dinesh1229/indian-sign-language-zklzt/models

[4] M. Oudah, A. Al-Naji, and J. Chahl, “Hand gesture recognition based on computer vision: a review of techniques”. Journal of Imaging, vol. 6, no. 8, pp. 73, 2020.

[5] H. Huang, Y. Chong, C. Nie, and S. Pan, “Hand gesture recognition with skin detection and deep learning method”. In Journal of Physics: Conference Series, vol. 1213, no. 2, pp. 022001,.2019, IOP Publishing.

[6] E. Alpaydin, Introduction to Machine Learning. Cambridge, MA, USA: MIT Press, 2020.

[7] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, Learning From Data, vol. 4. New York, NY, USA: AMLBook, 2012.

[8] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, pp. 436–444, May 2015.

[9] D. Konstantinidis, K. Dimitropoulos, and P. Daras, ‘‘A deep learning approach for analyzing video and skeletal features in sign language recognition,’’ in Proc. IEEE Int. Conf. Imag. Syst. Techn. (IST), Oct. 2018, pp. 1–6.

[10] Mohamed Hisham Jaward, Ming Jin Cheok, Zaid Omar. “A review of hand gesture and sign language recognition techniques.”. 2020.

[11] Tessa Verhoef, Patrick Boudreault, Oscar Koller. “Sign Language Recognition, Generation, and Translation: An interdisciplinary Perspective.”.2021.

[12] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans. “Sign Language Recognition Using Convolutional Neural Network.”.2022. Applications, vol. 3, August 2019.

[13] Starner T, Weaver J, Pentland A (1998) Real-time American sign language recognition using desk and wearable computer-based video. IEEE Trans Pattern Anal Mach Intel 20:1371–1375.

[14] Huang J, Zhou W, Li H, Li W (2015) Sign language recognition using 3D convolutional neural networks. In: IEEE international conference on multimedia and expo (ICME), pp 1-6.

[15] Huang J, Zhou W, Li H, Li W (2015) Sign language recognition using real-sense. In: IEEE China summit and international conference on signal and information processing (ChinaSIP), pp 166-170.

[16] Pigou L, Dieleman S, Kindermans PJ, Schrauwen B (2014) Sign language recognition using convolutional neural networks. In: Workshop at the European conference on computer vision. Springer, Cham, pp 572-578.

[17] Yang S, Zhu Q (2017) Video-based Chinese sign language recognition using convolutional neural network. In: IEEE 9th international conference on communication software and networks (ICCSN), pp 929-934.

[18] Bheda V, Radpour D (2017) Using deep convolutional networks for gesture recognition in American sign language. arXiv preprint arXiv:1710.06836.

[19] R.S. Sabeenian, S. Sai Bharathwaj, M Mohamed Aadhil. "Sign Language Recognition Using Deep Learining and Computer Vision.".2020.

[20] Tessa Verhoef, Patrick Boudreault, Oscar Koller. "Sign Language Recognition, Generation, and Translation: An interdisciplinary Perspective.".2021.

[21] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans. "Sign Language Recognition Using Convolutional Neural Network.".2022. Applications, vol. 3, August 2019.

[22] Badhe PC, Kulkarni V. Indian sign language translator using gesture recognition algorithm. In2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS) 2015 Nov 2 (pp. 195-200). IEEE.

[23] Lin HI, Hsu MH, Chen WK. Human hand gesture recognition using a convolution neural network. In2014 IEEE International Conference on Automation Science and Engineering (CASE) 2014 Aug 18 (pp. 1038-1043). IEEE

[24] Muhammad Al-Qurishi, Thariq Khalid, Riad Souissi. "Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues.". 2021 Sep 20.

[25] Ankita Wadhawan, Prateek Kumar. "Deep learning-based sign language recognition system for static signs.".2020 Jan 1.

[26] Melek Alaftekin, Ishak Pacal, Kenan Cicek. "Real-time sign language recognition based on YOLO algorithm.".2024 Feb 15.

[27] Soukaina Chraa Mesbahi, Mohamed Adnane Mahraz, Jamal Riffi, Hamid Tairi. "Hand Gesture Recognition Based on Various Deep Learning YOLO Models.". (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 14, No. 4, 2023