International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Real-Time Sign Language Recognition Using CNN and OpenCV for Inclusive Communication

B. Venkat Vivek[1], P. Chakradhar[2], S. Eswar Chaitanaya[3] and K. Dinesh Kumar[4]
Department of Artificial Intelligence and Machine Learning, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh, India

Abstract—Sign language is an indispensable medium of communication for individuals with hearing impairments, enabling them to interact and participate fully in society. However, the recognition and interpretation of sign language gestures present significant technical challenges due to their dynamic, spatially complex, and context-dependent nature. The "Real-Time Sign Language Recognition Using CNN and OpenCV" project aims to address these challenges through advanced machine learning and computer vision techniques, delivering an accessible, real-time gesture recognition system.

The system leverages a meticulously curated dataset comprising a wide range of sign language gestures captured in authentic environments. This dataset was sourced from publicly available repositories and augmented through custom video captures. Each gesture was recorded under diverse lighting conditions, varied backgrounds, and different signing styles to ensure robust generalization and adaptability. Using OpenCV for live video acquisition and preprocessing, the input data undergoes rigorous preparation before analysis. Central to the architecture is the implementation of Convolutional Neural Networks (CNNs), which efficiently extract both spatial and temporal features, enabling accurate and real-time gesture classification. To enhance performance, activation functions such as ReLU and SoftMax are integrated, alongside data augmentation strategies that ensure model adaptability to real-world conditions. This project aims to create an inclusive communication tool that bridges the gap for individuals unfamiliar with sign language, fostering accessibility and social equity.

I. INTRODUCTION

Sign language serves as a vital tool for the communication and expression of individuals with hearing impairments. Despite its importance, a significant barrier remains in enabling effective interaction between signers and non-signers. The complexities inherent to sign language—including its visual, spatial, and often dynamic characteristics—necessitate sophisticated approaches for automated recognition.

This project seeks to develop a scalable and efficient real-time recognition system using OpenCV for video capture and preprocessing, coupled with CNNs for gesture classification. By employing advanced machine learning methods, this system achieves high accuracy while remaining adaptive to diverse signing styles, environmental conditions, and linguistic nuances. Such a solution holds transformative potential in education, healthcare, and public communication domains, where accessibility and inclusivity are paramount.

II. BACKGROUND STUDY

Over the past two decades, substantial progress has been made in the field of sign language recognition (SLR), transitioning from hardware-dependent systems to vision-based approaches driven by artificial intelligence. Early methods relied on wearable devices, such as data gloves or sensors, which were effective but often impractical due to high costs and limited flexibility.

The rise of computer vision and deep learning technologies has shifted the focus to image and video-based methods. Convolutional Neural Networks (CNNs) have emerged as a cornerstone

for feature extraction and classification in SLR, demonstrating remarkable success in both static and dynamic gesture recognition. Studies utilizing pre-trained architectures, such as Alex Net and VGG16, have illustrated the efficacy of transfer learning in adapting models to SLR tasks. Additionally, real-time object detection frameworks like YOLO and SSD have been explored for their ability to achieve high-speed processing without compromising accuracy. YOLO (You Only Look Once) is renowned for its single-stage detection pipeline, enabling rapid inference by predicting bounding boxes and class probabilities simultaneously. Conversely, SSD (Single Shot Multibook Detector) combines multi-scale feature maps and anchor boxes to enhance detection precision for smaller objects. Both frameworks offer a balance of speed and accuracy, making them suitable for real-time applications such as sign language recognition.

While significant advancements have been achieved, challenges persist. Variability in signing styles, environmental noise, and background clutter necessitate robust preprocessing pipelines and diverse training datasets. Furthermore, integrating temporal modelling through recurrent neural networks (RNNs) or attention mechanisms remains an area of active exploration. This project builds upon these foundational works, employing a streamlined yet effective architecture that balances accuracy and computational efficiency.

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 depicts the suggested hand gesture recognition flowchart and the methodology that was Used. 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. The hand region is then extracted once the hand has been recognized and transferred to a better Yolo (You Only Look Once) deep learning model. 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. Then we labelled the dataset in roboflow [roboflow reference] tool present in the internet.

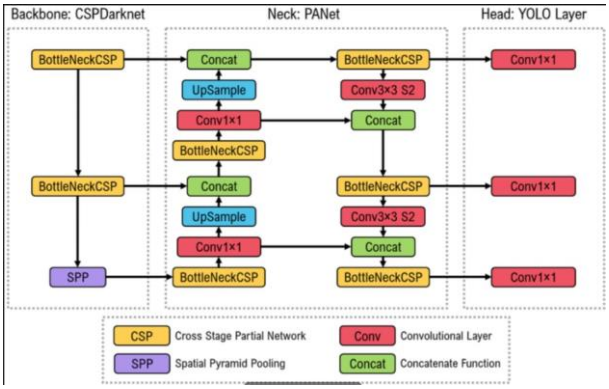


Fig. 1. Architecture of Yolo.

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 opensource 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 instances of each class from the dataset with the changes.

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 labeled. Before moving on to postprocessing, 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 analyze 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) Manual annotation: The annotation procedure, 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 manually drawing and categorizing bounding boxes, this tool makes it possible to change just one annotation or label throughout 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 bounding box around each object of interest in each image was necessary for the image annotation model, 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: A class label for each bounding box as well as one or more bounding boxes (each defined by a point, width, and height).

3) Image data labelling with bounding box: We have also produced a dataset with bounding box iffer ng so that we may utilize the characteristics of the deep learning detection technique. In order to reduce the difficulty and expense of iffer ng, 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 twodimensional

(2D) and threedimensional (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.

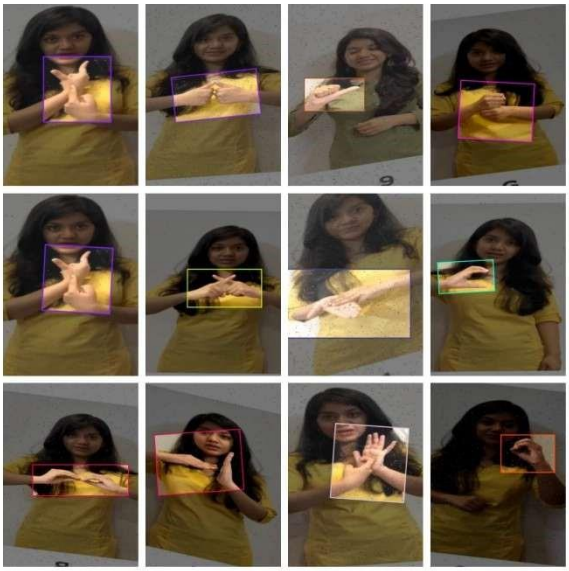


Fig. 3. Labeling different hand classes with bounding boxes.

D. Labeled Dataset

In a labelled dataset, each element of the unlabeled 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 unlabeled 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. The Structure of the Proposed YOLO Algorithms

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 makes use of only convolutional layers, to determine which items are represented in the image, a single fully convolutional network (FCN) is used. The YOLO method divides the image into cells or grids; each cell is responsible for object localisation, estimating the number of bounding boxes, and calculating 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. Labeling images is essential for good computer vision models. All the images are annotated and labeled 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. We label our dataset using YOLO annotation, which gives us certain values that are later detailed in the model process. 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 of collected images. The classes we have stored for our YOLO

2) YOLO v5 model: The Backbone, Neck, and Head architectural components of the YOLOv5 network are shown in Fig. 4. YOLOv5 Backbone: In order to extract features from images, including cross-stage partial networks, YOLOv5 uses CSPDarknet as its backbone. 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 [50]. 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. Compared to the other YOLO versions, it is substantially faster. CSPNET serves as the foundation for YOLOv5's feature map extraction from the image. In order to improve

information flow, it also makes use of the Path Aggregation Network (PANet) [51]. For the following reasons, we are utilizing YOLOv5 as it includes helpful elements like a cutting-edge activation function, a convenient manual, a hyperparameter, and a data augmentation technique. 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 differs from the Yolo series in several lighting areas:

1) Multiscale: utilize FPN to improve the feature extraction network rather than PAN [46], 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 [52]. It is more manageable and, in general, more cozy to utilize throughout training. Its architecture may be modified with equal ease, and it can be exported to numerous deployment environments [53].

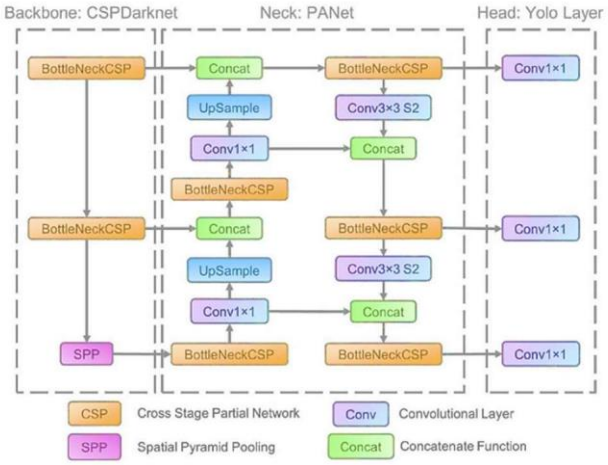


Fig. 4. The general architecture of the YOLOv5 network

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: This defines the number of images processed in a single training iteration. Larger batch sizes can speed up training but require more memory.

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

2. 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]

$$AP = \int P(R)\, dR \qquad (1)$$

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]

$$P = \frac{T}{T + F} \qquad (2)$$

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]

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

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]

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

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]

$$mAP = \frac{\sum AveP(q)}{Q} \qquad (5)$$

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.

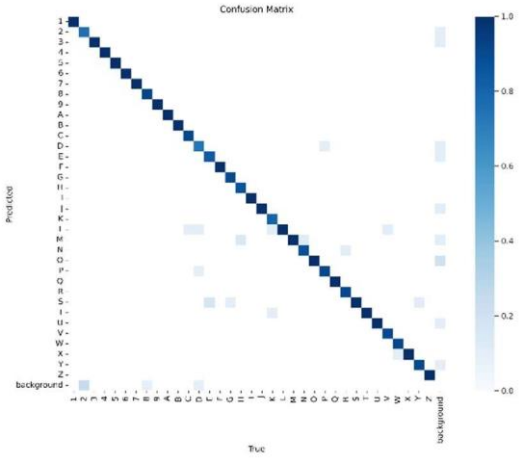
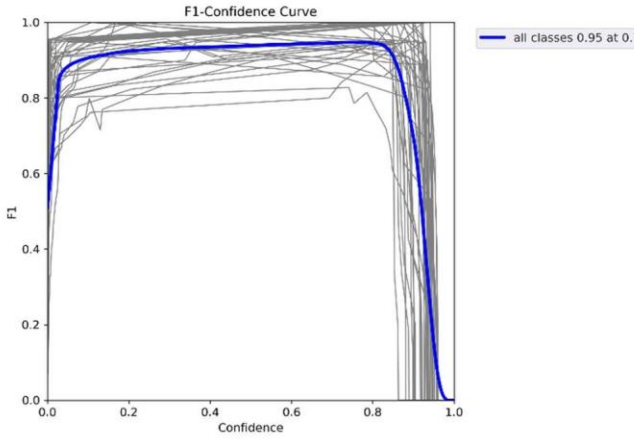


Fig. 6. Confusion metrics



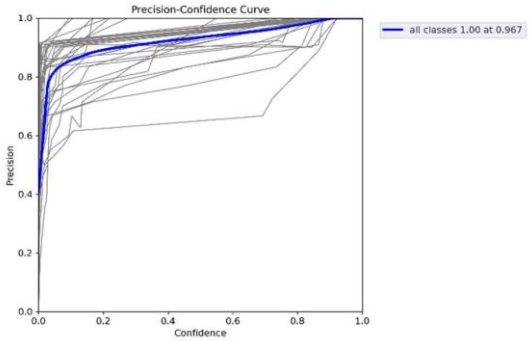


Fig. 8. Precision-Confidence Curve

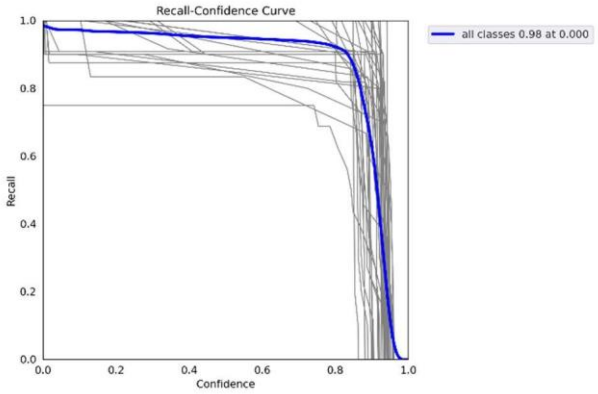


Fig. 9. Recall-Confidence Curve

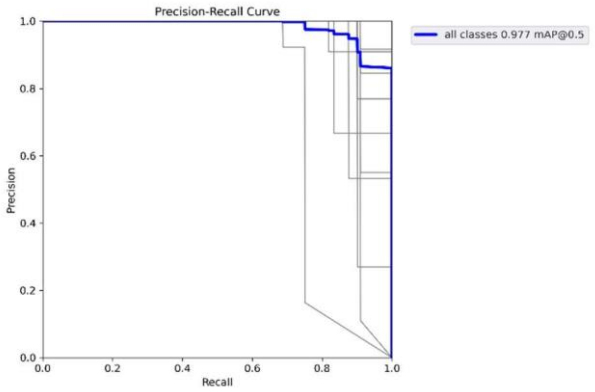
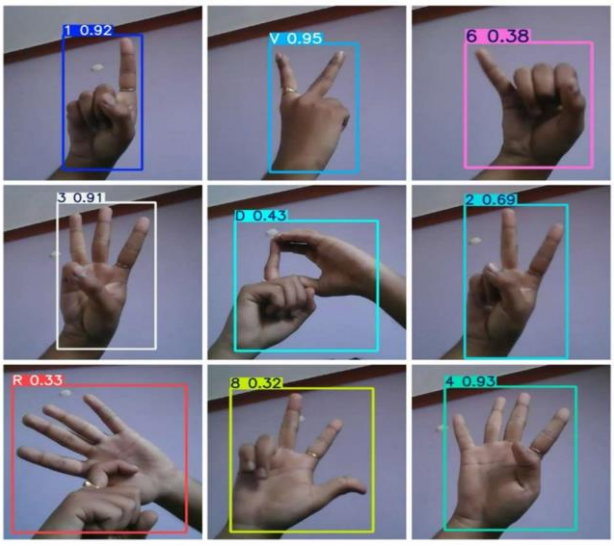


Fig. 10. Precision-Recall Curve



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+ Fig. 7. F1-Score / F1-Curve 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 https://docs.ultralytics.com/yolov5/

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

[3] 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.

[4] 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.

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

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

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

[8] 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.

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

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

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

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

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

[21] 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.

[22] 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

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

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

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

[26] 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