

Performance Comparison of Deep Learning Models for Real Time Sign Language Recognition

Irtika Khan¹, Sheetal Kumar², Pratham Rai³, Aarnav Rastogi⁴ and Leena Ragha⁵

¹⁻⁵Ramrao Adik Institute of Technology, D Y Patil Deemed to-be University, Navi Mumbai -400706, India

Abstract— Communicating with speaking/hearing impaired people has been one major issue throughout the years. Deaf and Dumb people use Sign Language consisting of various hand gestures as a means of communication. But, normal people fail to understand, as learning sign language is a tedious task and such language is required only in rare situations when we come across such people. Understanding what is conveyed is very important to convey back the message of doing the needful. There is a need for technology, which can reduce the problem of communicating with speaking/hearing impaired people. Survey shows an urgent requirement of robust real time sign language translators. So we aim to build a machine learning model which recognizes the hand gestures in real time through real time video processing and gives the meaning of the hand gesture as the textual output which can be further played as an audio signal. Through this research paper, we suggest the process to create dataset of Indian Sign Language and then using that dataset we have trained two models, YOLOv5 and YOLOv7 and carried out a comparative study on the same to learn about the differences in both models with respect to object detection, accuracy, precision, recall and more. The proposed YOLOv5 system attains accuracy of 87.6% while YOLOv7 system attains accuracy of 96.5% while being tested over locally created database for small phrases and sentences.

Index Terms— Object detection, Real time, Indian Sign language detection, OpenCV, YOLOv5, YOLOv7

I. INTRODUCTION

Communication is the act of conveying or sharing thoughts, sentiments, or information. In order for two individuals to communicate, they should be able to speak and comprehend the same language. Deaf and dumb persons, however, have been deprived of equal status in the society as they fail to express themselves to the normal people. The hearing/speaking impaired people use a different method of communication which is understandable by only those who have learnt it. They express themselves using visual sign language using body, face, and hand movements. This kind of people we come across rarely and find ourselves handicapped to communicate with them. If given a chance, they can live independently and can do wonders for themselves and for the society. Currently, there are more than 300 distinct sign languages throughout the world. Even though there are numerous distinct sign languages, only a small portion of the community is conversant in any of them, making it challenging for people with special needs to freely interact with the general public. Indian Sign Language Detection gives people a way to communicate in sign language even if they don't know it. Hence, there is a need for technology, that can address these issues in real time so that when there is a need we can use such translator tools to understand the sign language and also to communicate back to the Deaf and dumb

persons. Developing such solutions is a very challenging task as it involves the need for recognizing the sign language as it is being played, the numbers of gestures are large in number and they are formed using combinations of body, face, and hand movements. Similarly the other way round, playing the normal spoken language for deaf and dumb to make them understand what a normal person is saying is equally complicated. These applications need complex technology of video processing, audio processing, gesture recognition, Natural Language Processing, Speech Synthesis, etc, and a platform for the real time mobile application to make the dream of equality come true.

In this paper we propose to work on the challenge of developing an app that can process the real time continuous video frames to recognize the sign language and play the audio of the corresponding words and small highly used sentences. The paper presents the literature survey in section II, proposed solution in section III. Sections IV V and VI concentrate on the data set prepared for the experiment, preprocessing operations and the recognition system respectively. The results and analysis are presented in section VII and the section VIII gives the conclusion and future work of this research application.

II. PREVIOUSLY RELATED WORK

Towards this, we surveyed various technologies used in research works by various researchers and analysed the technologies for their capabilities, limitations, and the recognition accuracy.

In [1], ‘Sign Language Recognition and Converting into Text’, Tabassum and Raghavendra R(2022) built a human-computer interface to identify the 26 American sign language letters and produce text output that reflects the meaning of the letter. To identify and learn letters, the system employed CNN (Convolutional Neural Network) and LSTM (Long Short Term Memory) networks on custom made dataset and achieved 98% accuracy.

In [2], The authors developed a system that works with real-time input provided by the user. Support vector machines (SVM), convolutional neural networks (CNN), and other necessary techniques are used to recognise letters.

In [3], Convolutional Neural Networks were used by the authors to create a real-time system that combined image processing and machine learning to recognise 24 English alphabets of a custom dataset and achieved an accuracy of 83%.

In [4], the authors used neural networks to create a real-time method for finger spelling-based American Sign Language. Their method involves passing the hand through a filter first, then a classifier, which examines the category of hand motions. The model had a 96 percent accuracy rate for each letter.

In [5], multiple deep learning architectures for hand segmentation, local and global feature representations, and sequence feature globalisation and recognition were proposed by the authors as part of a system for dynamic hand gesture recognition. The King Saud University Saudi Sign Language (KSU-SSL) dataset of 40 dynamic hand motions was used to assess the proposed system, and they were able to attain an accuracy of 87.69%.

In [6], American sign language (ASL) motions are captured using a computer's webcam by the authors' desktop which then instantly translated them into text and voice. They employed CNN and achieved 95% accuracy in gesture identification.

In [7], the authors created system for speech and text translation from sign language in real time. They have created an Android application to show the vision-based method of converting sign language to text and voice without the need of any sensors and solely by photographing the hand motions. In the system proposed by them, they used SVM [Support Vector Machine] and Principal Component Analysis model is used to identify 26 English alphabets along with the first 10 numerical digits in dataset consisting of 1000 samples per symbol. They have quoted an accuracy of 92%.

In [8], through a gadget that makes use of an Arduino Uno board, a few flex sensors, and an Android application to promote user involvement, the author hopes to address this particular issue of communication with hearing/speaking impaired people. The wearer's motions and gestures are detected by the flex sensors, and depending on the parameters for the various values produced, the appropriate messages are delivered through a Global System for Mobile (GSM) Module to the user's Android handset, which converts the text message to speech.

In [9], the study used MATLAB to demonstrate sign language recognition of 26 Indian sign language hand movements. The suggested system had four modules, including feature extraction, sign recognition, sign to text, and pre-processing and hand segmentation. The gesture was recognised using the Linear Discriminant Analysis (LDA) technique, and the detected gesture was then transformed to text and speech format.

In [10], The authors created a system for the real-time definition of hand gestures using a registered neural network framework and network weights. The system uses a convex hull algorithm to determine the skin colour for a

specific frame for hand use, the convex hull algorithm to determine the hand gesture, and finally the registered neural network system and network weights. The real-time system has a 98.05% accuracy rate.

In [11], The authors described a technique that used support vector machines (SVM) and artificial neural networks to translate Bengali Sign language into text. The hand gesture made in front of the camera is the input, which is captured by Microsoft Kinect. Once joints and wrists have been identified and the shapes have been evaluated, the collected hand sign is finally recognised. To classify the sign, a contour feature is taken and processed through an SVM..They achieved an accuracy of 84.11%.

We observed that, while Neural Networks have been used successfully for Sign Language letter recognition in the past, majority of those need a three-dimensional capture component with motion-tracking gloves or a Microsoft Kinect, which limits the scalability and feasibility of these solutions. There are currently relatively few goods on the market that assist deaf individuals in interacting with hearing people. Existing items are difficult to get. ASL Translator and iCommunicator can only convert English words into sign language video representations. So yet, there are just a few sign language items available. The apps that are currently available are for learning sign languages and don't have the facility of real time conversion of sign into plain text. Due to these challenges, it is difficult to build and find an efficient application that converts live input of signs into text and has a real time functionality.

III. PROPOSED SYSTEM

The figure given below, Fig.1, represents block diagram showing the working of a real time sign language detection system. As shown, the proposed system takes the real time video as the input, extracts the frames, detects the hand and tracks it in the real time video. It then segments the hand portion, performs pre-processing and uses the pre-processed image for the wanted regions of the hand which is then used for the sign recognition. The recognized output can be displayed on the screen or can be played as audio. These steps are further explained in detail in the following subsections.

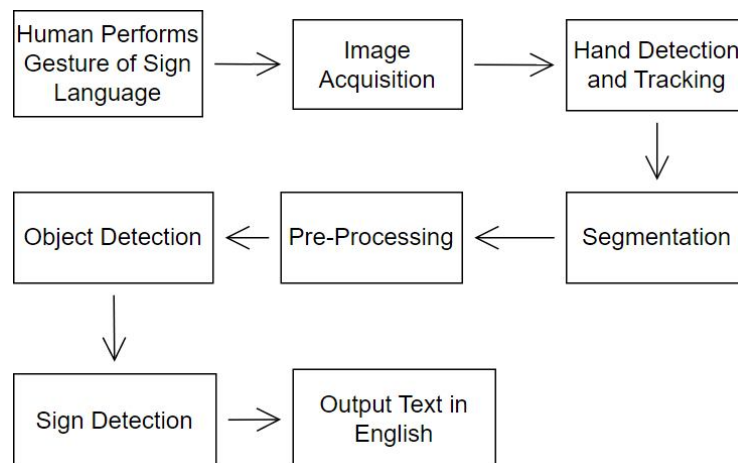


Fig.1. Block Diagram Representing working of Real Time Sign Language Detection System

IV. DATASET

Through the survey, it is observed that, although many data sets are available, they are mainly for alphabets and numbers. Also, it is observed that the sign language varies from region to region. Hence we chose to work on creating our own database which consists of common phrases and words in the Indian Sign Language as there are very limited applications which support ISL. For creating the dataset videos of each person was captures and images frames were extracted using the time module in python.

Time module in Python allows for the representation of time in code using a range of different types, including objects, integers, and texts. Time can also be used to wait while executing code or to assess the effectiveness of the code. It is utilized in this instance to introduce pauses between shots to allow for motions. The names of the picture files were generated using the uuid library. 128-bit ids help create random objects while ensuring uniqueness because they are generated based on time and computer hardware.

The existing data sets majorly concentrated on sign languages of alphabets. As we wanted to work on real time requirements of understanding the phrases and sentences conveyed by the deaf and dumb persons, for the

experiments, we created our own data sets. The generated dataset consists of 20 gestures for the following phrases and short sentences from the Indian Sign Language.

1. Bright	2. House	3. Loan	4. Thanks
5. Child	6. How Are You	7. Love	8. Toilet
9. Difficult	10. Indian	11. Namaste	12. Understand
13. Good	14. What Is The	15. Sorry	16. Vegetarian
	Time		Food
17. Hello	18. Where	19. Teacher	20. Water

For each gesture, we prepared 12 videos of 12 different people from which later the frames were extracted. We extracted these images from videos in real time using tools developed using openCV and python and stored in their respective folders. This way we created a dataset consisting of a total of 1746 images for our experiments.

V. PRE PROCESSING

As we created our own dataset, each image has its own issues and so we performed extensive preprocessing operations on them to make them suitable for feature extraction.

Pre-processing is performed using the open CV tools. The operations performed in the sequence are:

- Orientation correction,
- Isolate Objects- segmentation
- Background Elimination,
- Flip: Horizontal and Vertical,
- Rotation: Between -17° and $+17^\circ$,
- Shear adjusted between $\pm 15^\circ$ Horizontal, $\pm 12^\circ$ Vertical,
- Brightness adjusted between -25% and +25%, Exposure between -25% and +25%
- Static Crop between 2-48%
- Horizontal Region between 9-75%
- Vertical Region Auto-Adjust
- Contrast enhancement using Adaptive Equalization
- Gray scale conversion: Applied

In preprocessing, we first extracted the images from the videos present in the dataset to create a more comprehensive dataset with all image data points. We checked the orientation of all data points to ensure there were no dissimilarities. We then isolated the objects which in our case is the sign being performed and eliminated the background noise present. Then, we performed standard augmentation steps to increase the diversity in the dataset by applying flip, rotation shear and brightness. Then this newly augmented data was put under static crop as well as horizontal and vertical adjustment to fit the frame. Lastly we applied contrast enhancement and grayscale to overcome any problems that may arise due to the camera quality, light reflections, shadows or any such real time dependencies.

VI. RECOGNITION SYSTEM.

For object detection in the given image, we proposed to use YOLO which stands for "You Only Look Once". We chose to use the versions of YOLO technologies as they are popular real- time object detection system used in computer vision applications. Here's a step-by-step explanation of how YOLO works for image recognition:

a. The input image is divided into a grid of cells. Each cell has a fixed size and aspect ratio. For each cell, YOLO predicts a fixed number of bounding boxes (usually 2-3). Each bounding box is defined by four parameters: x and y (the centre of the box), width, and height. For each bounding box, YOLO predicts the probability that the box contains an object and the class of the object. YOLO uses a softmax function to produce class probabilities for each bounding box. YOLO applies non-max suppression to eliminate overlapping bounding boxes with lower confidence scores. Only the highest-confidence bounding box is kept for each object. The final output of YOLO is a set of bounding boxes, each with its corresponding class and confidence score. YOLO can be trained on large datasets such as COCO or VOC to learn to recognize different classes of objects. During training, the model adjusts the weights of its layers to minimise the difference between predicted and actual bounding boxes. In order to improve the accuracy and speed of YOLO, various modifications have been proposed, such as YOLOv2 and YOLOv3, which use different architectures and training techniques.

b. The major update that has been done is to change the number of classes which was initially 7, has been updated to 20 signs . A total of 1746 data points were added in a dataset and the model will be trained on the same. After

setting up and updating the configuration, the model was trained in 10000 steps. After training, the model is loaded from the latest checkpoint which makes it ready for real-time detection.

c. After setting up and updating the configuration, the model will be ready for training. The trained model is loaded from the latest checkpoint which is created during the training of the model. This completes the model making it ready for real-time sign language detection.

Following the hand gesture data creation for the gesture recognition experiments, a labelled chart is created which is a representation of all the objects within the model, i.e., it contains the marker of each sign along with their id. The id will be used as a reference to look up the class name and all the information and path names are saved in a data.yaml file.

VII. RESULTS

A. Setup For The Experiment.

The dataset is created using Indian Sign Language where the signs are common phrases used in our daily lives. The dataset is created using the data acquisition method mentioned in Dataset section IV. This experimentation was carried out on a system with Windows 10 and above or a macOS BigSur 11.6.4 or later along with an Intel HD Graphics 6000 1536 MB graphics and 1.8 GHz Dual-Core Intel Core i5 processor. We also need a HP TrueVision HD camera with 0.31 MP and 640x480 resolution and programming environments such as OpenCV, Google Colab, GitHub API and Ultralytics API. OpenCV primarily provides functionality for real-time computer vision. It accelerates the use of machine vision in commercial products and provides common infrastructure for computer vision-based applications.

B. Implementation Details

1. For YOLOv5 Model

The same team that created the original YOLO algorithm released YOLO v5 in 2020 as an open-source project, which is now maintained by Ultralytics. The success of earlier iterations is built upon by YOLO v5, which also includes a number of enhancements and new features. In contrast to YOLO, YOLO v5 uses a more intricate architecture called EfficientDet, which is based on the EfficientNet network architecture (architecture illustrated above). YOLO v5 can achieve greater accuracy and better generalization to a larger variety of item categories because to the use of a more complicated architecture. The training data used to develop the object identification model differs between YOLO and YOLO v5. The PASCAL VOC dataset, which has 20 different object categories, was used to train YOLO. using the other hand, YOLO v5 was trained using D5, a larger and more varied dataset that consists of a total of 600 object types. The anchor boxes are created using a new technique in YOLO v5 called "dynamic anchor boxes." The ground truth bounding boxes are first grouped into clusters using a clustering method, and then the centroids of those clusters are used as the anchor boxes. As a result, the anchor boxes can match the size and shape of the identified objects more closely. The idea of "spatial pyramid pooling" (SPP), a kind of pooling layer used to lower the spatial resolution of the feature maps, is also introduced in YOLO v5. Since SPP enables the model to view the objects at various scales, it is employed to enhance the detection performance for small objects. SPP is used by YOLO v4 as well, however YOLO v5 makes a number of changes to the SPP design that enable it to perform better. Google Colab, which offers no-cost access to potent GPUs and doesn't require configuration, was used to train the model. We used a Roboflow.ai notebook that is built on YOLOv5 and employs pre-trained COCO weights. In order to train the upper layers of the model to recognize our classes, we included our dataset for Indian Sign Language and changed the number of training epochs as well as the stack size. It requires roughly 120 minutes to train a model for 250 epochs. Figure 2's graphs, which show several performance measures for both the training and validation sets, show how our model has improved. Fig. 2 displays three different types of loss for the training and validation sets: classification loss, objectness loss, and box loss. The X-Axis displays the data point's epoch number, and the Y-Axis displays a percentage value in decimal form.

The box loss measures how well an object's center can be located by the algorithm and how well a predicted bounding box encloses an The mean average precision (mAP), which changes over the training epochs and achieves a value of 87.6% after the item, is shown in Figure 3. For the YOLOv5 model, we can see that the box loss value falls to less than 1% for training and less than 2% for validation after training for 250 epochs. The probability that an object exists in a suggested zone of interest is basically measured by objectness. For the YOLOv5 model, we can see that the objectness loss value drops to less than 0.6% for both training and validation after 250 epochs of training. If the objectivity is high, an item is probably present in the image window.

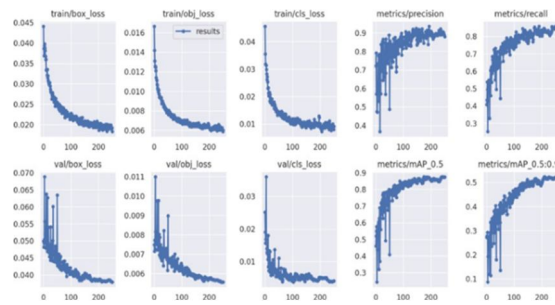


Fig. 2. Plots for the YOLOv5 model's training and validation set for box loss, objectness loss, classification loss, precision, recall, and mAP over the training epochs.

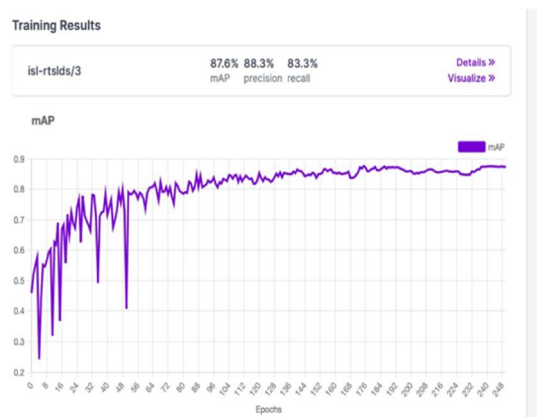


Fig.3. Graph showing mean average precision (mAP) for YOLOv5 model

How successfully the algorithm can determine the proper class of a given object is shown by the classification loss. For our YOLOv5 model, we can see that the value of classification loss reduces to less than 1% for both training and validation after training for 250 epochs.

We can also see the plots of recall and precision in Figure 2. A model's precision at the time of prediction is measured by its precision. All guessed positives are subtracted from the true positives. We can observe that 88% precision was attained for the YOLOv5 model. The recall serves as a gauge of how well our model detects True Positives. A 83% recall rate was attained. The mean Average Precision with an IoU (Intersection over Union) of 0.50, or 50%, is designated as mAP_0.5. The amount of a predicted bounding box that overlaps with the ground truth bounding box, divided by the combined area of both bounding boxes, is used to calculate intersection over Union (Iowa). The mean Average Precision (mAP_0.5:0.95) has an IoU interval of 0.50 to 0.95.

2. For YOLOv7 Model.

The depth of the model is scaled up in the architectural diagram, which raises the quantity of convolutions. It is necessary to scale the model's depth before scaling the transmission layer with the relevant width scaling factor. For the purpose of identifying objects of various shapes, anchor boxes are a collection of preconfigured boxes with various aspect ratios. With nine anchor boxes, YOLO v7 can detect a larger variety of item forms and sizes than earlier iterations, which helps to lessen the amount of false positives. The implementation of a new loss function termed "focal loss" in YOLO v7 is a significant improvement. Standard cross-entropy loss function, which is known to be less successful at identifying small objects, was utilized in earlier iterations of YOLO. By de-weighting the loss for well-classified samples and concentrating on the challenging examples—the difficult-to-detect objects—focal loss combats this problem. Additionally, YOLO v7 has a greater resolution than earlier iterations. Compared to YOLO v3, which processed photographs at a resolution of 416 by 416 pixels, it processes images at a level of 608 by 608 pixels. YOLO v7 can detect tiny objects with greater precision thanks to its better resolution.

The quickness of YOLO v7 is one of its key benefits. 155 frames per second is substantially faster than other cutting-edge object detection systems in processing images. Even the baseline YOLO model could process at a top speed of 45 frames per second. This qualifies it for delicate real-time applications like surveillance and autonomous vehicles, where faster processing speeds are essential.

The input was convolved with the 2 and 4 blocks of the 3x3 convolution using the same channel multiplier, and two more connections to the block were made. The input is fed directly into the bottom block using 1x1 convolution. At the base of the block, all of the features are concatenated, and 1x1 convolution is applied to extract rich information. When building a successful network, some properties, such as the number of calculations and computational density, are typically optimized. The YOLOv7 architecture is built on the ELAN (efficient layer aggregation network) technology. The shortest and longest gradient paths are managed by ELAN to build an efficient network, which is necessary for deeper networks to converge and train effectively. ELAN's modules are shown in the diagram below. The input was convolved with 2 and 4 blocks of 3x3 convolution using the same channel multiplier to create two more connections to the block, but it is fed directly into the bottom block using 1x1 convolution. Towards the block's base, while building an efficient network, the amount of computations, computational density, and a few other variables are typically tuned. The YOLOv7 architecture is based on ELAN (efficient layer aggregation network). ELAN considers building an efficient network by managing the shortest and longest gradient paths in order to let deeper networks converge and train effectively. The ELAN modules are shown in the diagram. The input was convolved with the 2 and 4 blocks of the 3x3 convolution using the same channel multiplier, and two more connections to the block were made. The input is fed directly into the bottom block using 1x1 convolution. At the base of the block, all of the features are concatenated, and 1x1 convolution is applied to extract rich information. Fig.4. displays three different types of loss for the training and validation sets: classification loss, objectness loss, and box loss. For the YOLOv7 model, we can see that the box loss value falls to less than 1% for both training and validation after 200 epochs of training. The value of classification loss drops to less than 0% for both training and validation after 200 training epochs, and the value of objectness loss decreases to less than 0.5%.

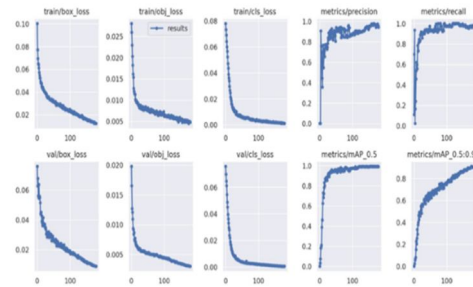


Fig. 4. Plots of box loss, objectness loss, classification loss, precision, recall and mean average precision (mAP) over the training epochs for the training and validation set for YOLOv7 model.

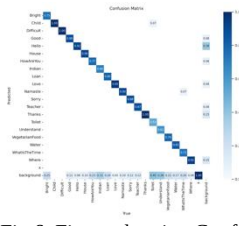
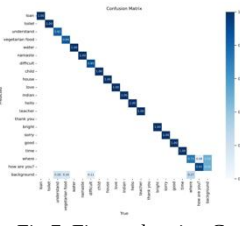
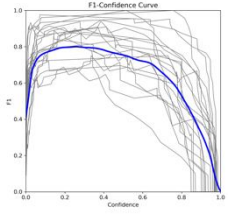
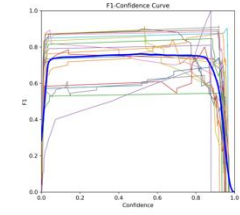
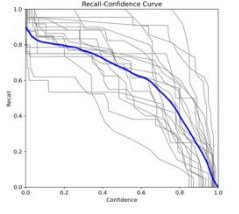
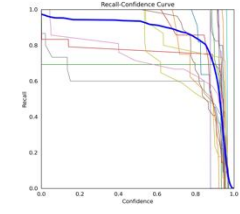
The developed models were able to detect real time sign language. The system was trained using the custom dataset created which consisted of 20 signs of Indian Sign Language which were: bright, child, difficult, good, hello, how are you, house, India, loan, love, Namaste, sorry, teacher, thanks, toilet, understand, vegetarian food, water, what is the time, where. The output video frames of the real time test environment of various phrases are shown in Fig. 5 with text showing the result of recognition.

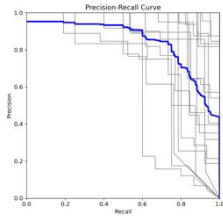
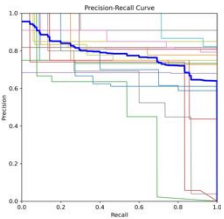
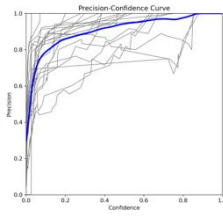
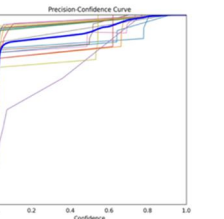


Fig.5 Detection of 20 gestures of the ISL (Indian Sign Language) in real time

C. Result Analysis

TABLE I. A COMPARATIVE STUDY BETWEEN YOLOV5 AND YOLOV7.

Parameters	YOLOv5	YOLOv7
Accuracy	The YOLOv5 model has an accuracy of 87.6%	The accuracy we achieved is 96.5%
Precision	The precision of our model is 88.3%	The precision of our model is 87.5%
Recall	The recall of our model is 83.3%	The recall of our model is 96.6%.
Confusion Matrix	We have added a 21st class x in order to confuse the system and increase precision.	We didn't add an additional empty class; the performance difference was un-noticeable
Data Pre-processing	We have added various pre-processing and augmentation steps to our raw dataset and trained the model on a total of 8000 data points.	We have not carried out any data pre-processing.
FPS	The signs are getting detected at the minimum of 4fps	The signs are detected properly via video format and real-time at 3fps
Signs detected	Almost all 20 signs are detected, each having a relatively high accuracy.	Over 19 signs are getting perfectly detected, others are showing low confidence
Compatibility	YOLOv5 is compatible with both android and iOS applications.	YOLOv7 is only compatible with iOS however it causes overheating.
Confusion Matrix	 Fig.6. Figure showing Confusion Matrix of YOLOv5 model.	 Fig.7. Figure showing Confusion Matrix of YOLOv7 model
F1 confidence	 Fig.8. Figure showing average F1 confidence: all classes 0.8 at 0.258	 Fig.9. Figure showing average F1 confidence: all classes 0.76 at 0.513
Recall-Confidence curve	 Fig.10. Figure showing Recall-Confidence:all classes 0.90 at 0.0	 Fig.11. Figure showing Recall-Confidence:all classes 0.97 at 0.0

Precision recall curve	 Fig.12. Figure showing Precision-Recall curve: all classes 0.84 mAP@0.5	 Fig.13. Figure showing Precision-Recall curve: all classes 0.77 mAP@0.5
Precision confidence	 Fig.14. Figure showing Precision-Confidence curve: all classes 1 at 0.906	 Fig.15. Figure showing Precision-Confidence curve: all classes 1 at 0.994

VIII. CONCLUSIONS AND FUTURE WORK

Sign language detection is important as we are not able to communicate with deaf and mute people and real time sign conversion systems with good accuracy and performance for Indian Sign Language Detection were not available. In the systems developed, sign detection of gestures Indian Sign Language can be easily done in real time using YOLO. The systems have been trained to recognize simple signs of sign language such as Bright, Child, Difficult, Good, Hello, how are you, Loan, Love, Home, Where, Water, Understand, Indian, what is the time, Namaste, Toilet, Vegetarian Food, Teacher, Sorry and Thank you. As the systems detect these signs in real time, the communications barrier between the disabled and abled can be easily crossed. This study compared and contrasted the YOLOv5 and YOLOv7 object detection models on 20 gestures of Indian Sign Language dataset. As shown in Fig 17, the YOLOv5 and YOLOv7 models were compared in terms of Accuracy, Precision, Recall, F1-confidence, Recall-Confidence Values, Precision Confidence values, Precision-Recall values as well as other factors like signs detected, frames per second and compatibility. We conclude that the YOLOv7 performs better in all respects and is also time efficient for real time applications.

For future work, we plan on enlarging our dataset so as to improve accuracy as well as increase the number of signs the systems can recognize. We also plan to make the dataset more diverse and include the sign languages used in various regions of India and the world. The systems would then be able to recognize regional and local languages as well. We also plan to add the functionality of prediction of the next gesture using Natural Language Processing for faster and better interpretation. This system can also be used as an API with various virtual assistants like Apple Siri and Amazon Alexa in the future.

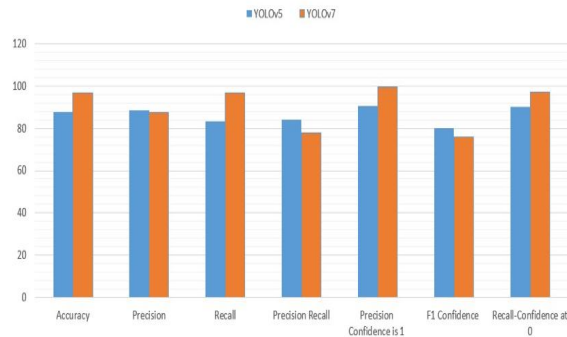


Fig.16. Figure showing comparison of various parameters for YOLOv5 and YOLOv7 models.

REFERENCES

- [1] Shaheen Tabassum, Raghavendra R, Sign Language Recognition and Converting into Text, International Journal for Research in Applied Science & Engineering Technology (IJRASET) Volume 10 Issue IV Apr 2022
- [2] Amit Dighe, Suraj Adsul, Saurabh Wankhede, Sonam Borhade, Sign Language Recognition Application using Py-thon and OpenCV, International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 2, Issue 3, May 2022
- [3] J. J. Raval and R. Gajjar, "Real-time Sign Language Recognition using Computer Vision," 2021 3rd International Conference on Signal Processing and Communication (ICPSC), Coimbatore, India, 2021, pp. 542-546, doi: 10.1109/ICSPC51351.2021.9451709
- [4] Vatsal Patel, Maahi Patel, Sign To Text Conversion- Helping Aid International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2021
- [5] M. Al-Hammadi et al., "Deep Learning-Based Approach for Sign Language Gesture Recognition With Efficient Hand Gesture Representation," in IEEE Access, vol. 8, pp. 192527-192542, 2020, doi: 10.1109/ACCESS.2020.3032140.
- [6] Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur, Dr. Dayananda P, Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network, International Journal of Engineering Research & Technology (IJERT), 2020
- [7] K. Tiku, J. Maloo, A. Ramesh and I. R., "Real-time Conversion of Sign Language to Text and Speech," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 346-351.
- [8] Bharath A Manoj, Conversion of Sign Language to Text and Speech and Prediction of Gesture, International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-6, March 2020
- [9] Mahesh Kumar N B, Conversion of Sign Language into Text, International Journal of Applied Engineering Research, Volume 13, Number 9 (2018) pp. 7154-7161.
- [10] M. Taskiran, M. Killioglu and N. Kahraman, "A Real-Time System for Recognition of American Sign Language by using Deep Learning," 2018 41st International Conference on Telecommunications and Signal Processing (TSP), Athens, Greece, 2018, pp. 1-5, doi: 10.1109/TSP.2018.8441304.
- [11] A. R. Chowdhury, A. Biswas, S. M. F. Hasan, T. M. Rahman and J. Uddin, Bengali Sign language to text conversion using artificial neural network and support vector machine, 2017 3rd International Conference on Electrical Information and Communication Technology (EICT), 2017, pp. 1-4.