VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

MACHINE VISION

BCSE417L

ASSIGNMENT FINAL REPORT

TOPIC: SIGN LANGUAGE TRANSLATION

SUBMITTED TO

Dr. G Bharadwaja Kumar Sir

SUBMITTED BY

KAPILESH N – 22BAI1459

ABSTRACT:

- This project aims to make people who are born with disabilities like being mute or hard of hearing mingle with everyone across the world.
- It aims to translate American Sign Language (ASL) to its commonly readable English form.
- This not only helps the disabled communicate with a layman but also may potentially help people learn ASL either to support a family member or a obtaining a disability themselves.

INTRODUCTION:

- ASL is a very commonly used sign language used all over the world to communicate with or between PWDs (People with Disabilities)
- A translator that converts the hand signs live into their English meaning would be a great step towards helping the disabled community navigate life and help them reach an equal footing despite their shortcomings.
- It could also help people who have lost their hearing or speech learn sign language and gain fluency in a crucial skill.
- Here, we’ve used a YOLOv5 based model to help detect various ASL signs and display them in plain English.
- A self-made dataset was created to train the model and then apply transfer learning using pytorch to get a working model.

PROBLEM STATEMENT:

The objective of this project was to be build a fully fledged Sign Language capable of running off a laptop webcam or any other equally low resolution camera to help PwDs communicate with laymen and others who may not know Sign Language. A model needs to be trained to recognize all signs and should be able to implement in real time using the live footage from a webcam.

METHODOLOGY:

1. **Data Collection & Annotation**
   - **Gesture Selection**: Focus on commonly used ASL gestures such as "Hello," "Yes," "No," "Thanks," "I love you," and "Please". (Due to training time and data collection constraints)
   - **Image Acquisition**: Capture images or video frames of these gestures under varying lighting conditions and backgrounds to ensure robustness. 865 images are present in total.
   - **Annotation**: Utilize tools like RoboFlow to draw bounding boxes

around hand gestures in each image, assigning appropriate class labels. Annotations are saved in YOLOv5 PyTorch format for compatibility.

2. **Model Training with YOLOv5**
    - **Model Selection**: Choose an appropriate YOLOv5 variant (e.g., YOLOv5s for speed or YOLOv5m for a balance between speed and accuracy). We went with YOLOv5s for quicker real-time translation
    - **Transfer Learning**: Leverage pre-trained weights on large datasets like COCO to initialize the model, facilitating faster convergence and improved accuracy. This was available in the
    - **Training Configuration**:
        - **Image Size**: Typically set to 640x640 pixels.
        - **Batch Size**: Determined based on available GPU memory. Here, batch size of 16 was used.
        - **Epochs**: Train for sufficient epochs (e.g., 50-200) to ensure convergence. In our case, the model was trained for 500 epochs.
        - **Data Augmentation**: Shearing and rotation of ± 15° both. This was done during the annotation process and the data downloaded was already augmented.

3. **Real-Time Inference**
    - **Input Processing**: Capture frames from a webcam or video feed.
    - **Detection**: Pass each frame through the trained YOLOv5 model to detect and classify hand gestures using the weights from the trained model.
    - **Output Rendering**: Overlay bounding boxes and class labels on detected gestures within the frame.
    - **Text Translation**: Convert detected gestures into corresponding text and display with the bounding box.

ARCHITECTURE:

Overall, the model has the following attributes;

157 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs

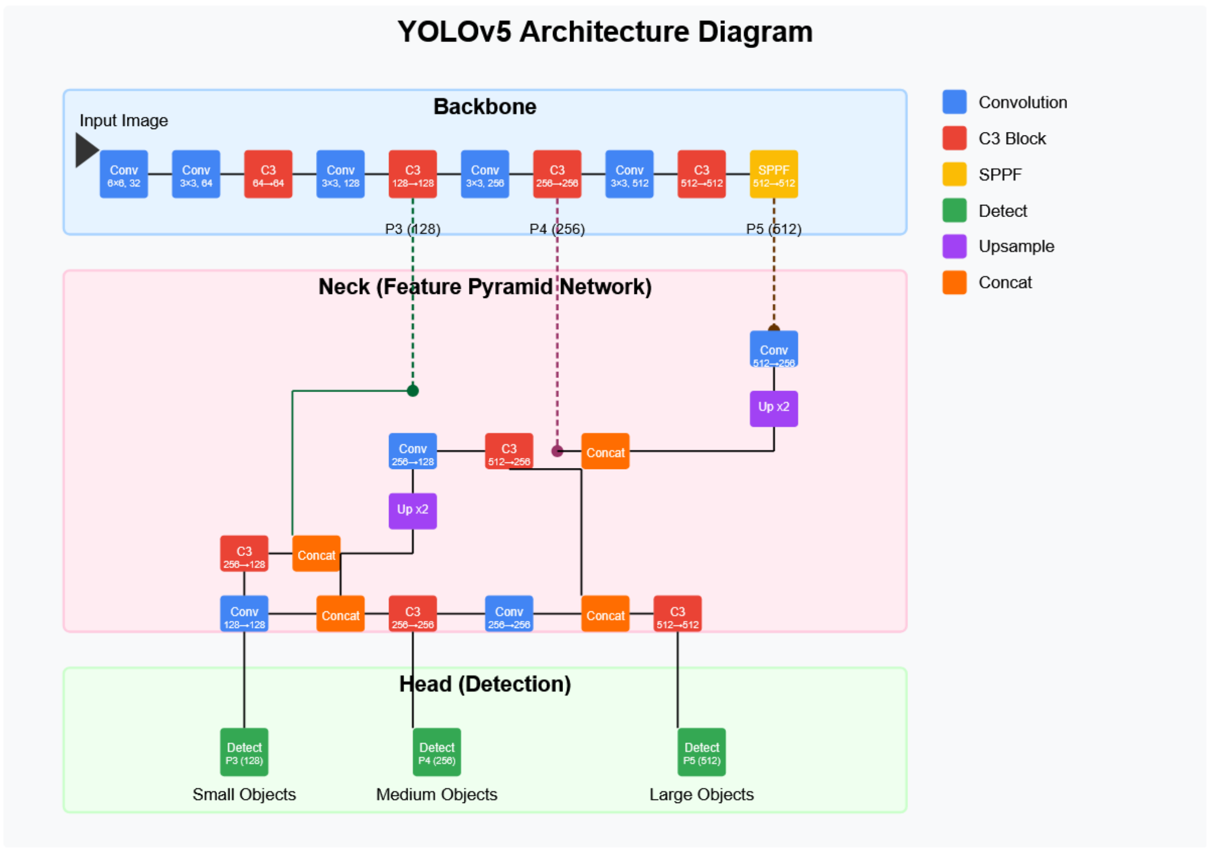
In more detail,

- **Backbone**: The initial part of the network that extracts features from the input image
    - Starting with Conv layers (6×6 and 3×3)
    - Multiple C3 blocks (Cross Stage Partial Networks) for feature extraction

    - Gradually increasing channels from 32→64→128→256→512
    - Ending with SPPF (Spatial Pyramid Pooling - Fast) for enlarged receptive field
- **Neck**: Feature Pyramid Network (FPN) that creates multi-scale feature maps
    - Upsampling paths that increase spatial resolution
    - Concatenation operations that merge features from different scales
    - Additional C3 blocks for feature refinement
    - Creates feature maps at three different scales (P3, P4, P5)

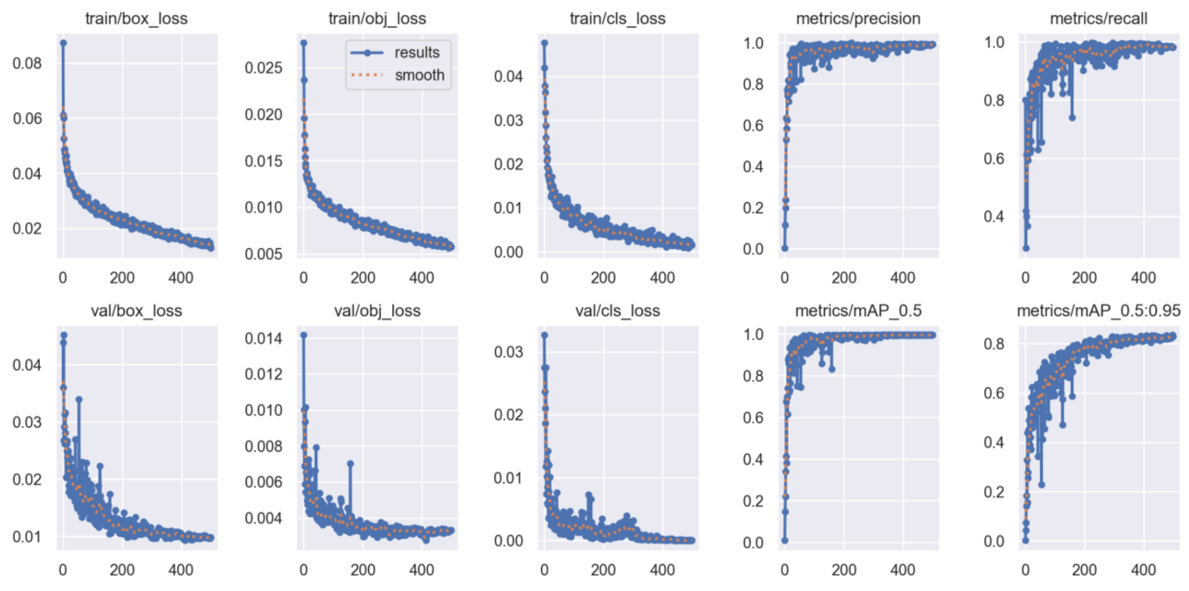
- **Detection Head**: Final layers that produce bounding box predictions
    - Three detection heads operating at different scales
    - P3 for small objects (higher resolution)
    - P4 for medium objects
    - P5 for large objects (lower resolution)

The condensed version of the architecture is as shown below

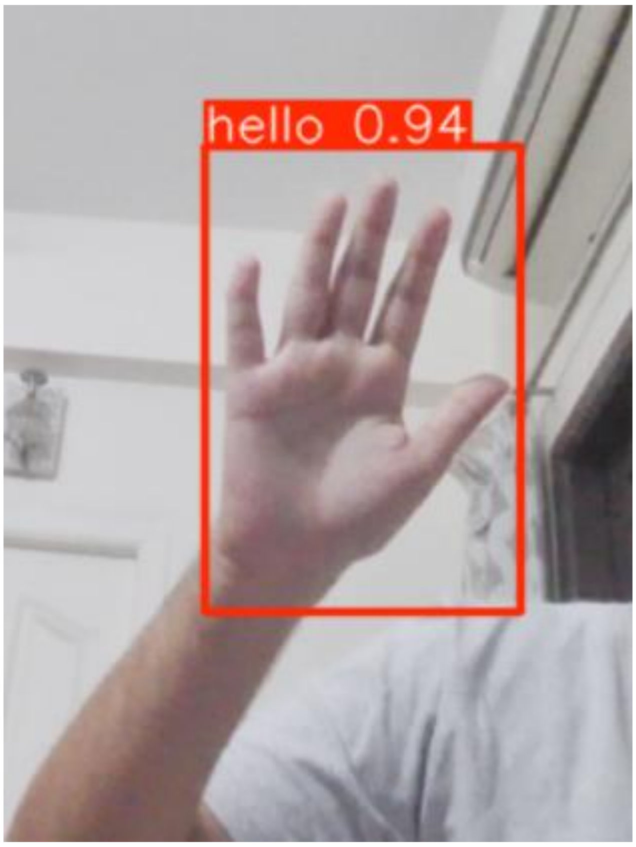


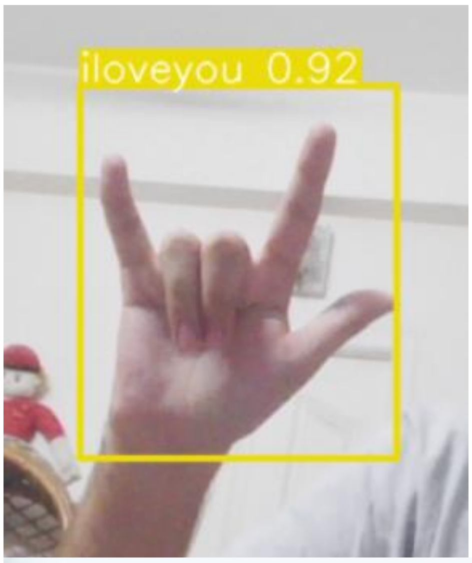
RESULTS:

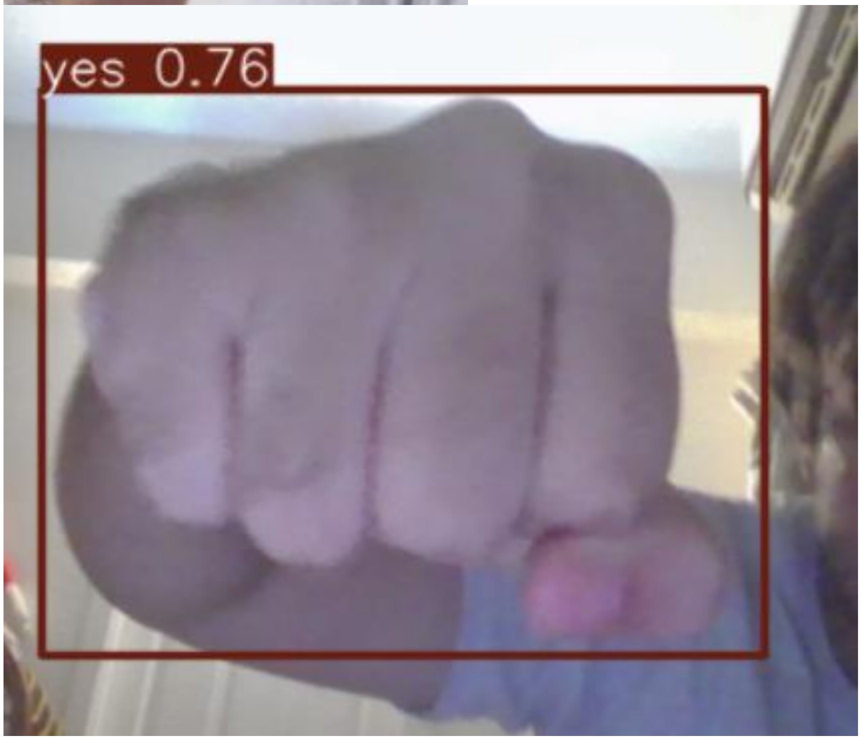
These are the losses, performance and accuracy metrics are

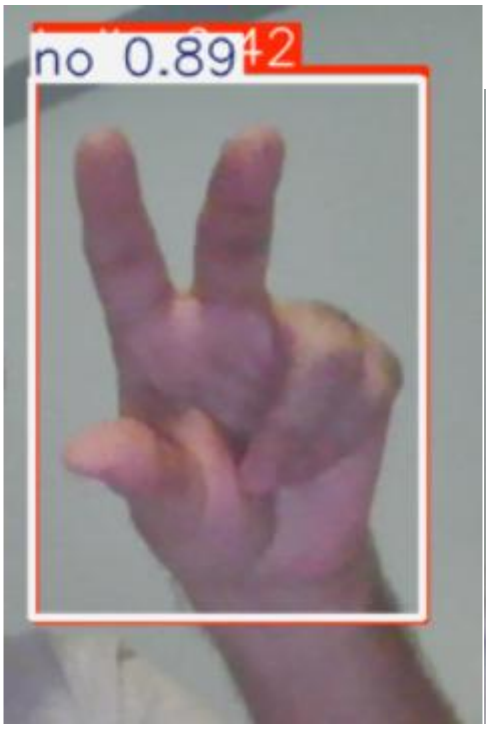
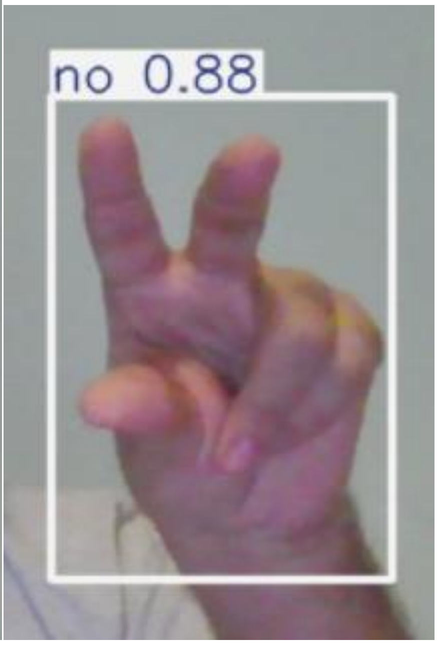
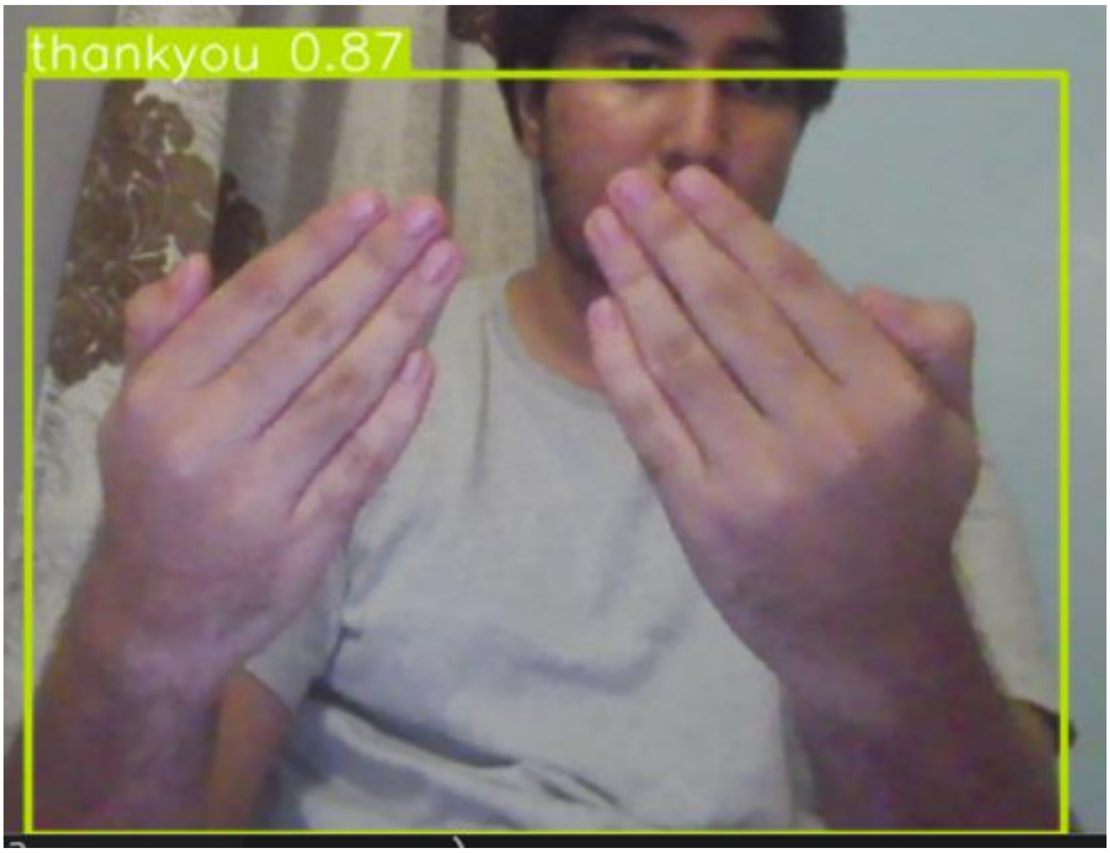


The following are live examples of how the model preforms using my laptop webcam




iloveyou 0.92


yes 0.76


no 0.89
42

no 0.88

thankyou 0.87

COMPARISON WITH OTHER MODELS:

Modeling efficient YOLOv5-based achieves high accuracy for sign language letters with real-time performance, powering accuracy: 95% precision, 97% recall, and 96% mAP@0.5. It is very efficient for devices with limited computational resources like a smartphone. An immediate feedback application would benefit very much from this good balance between accuracy and speed.

YOLOv8-Based Models with higher metrics like 98% precision/recall as opposed to both percentages in YOLOv5 and a 99% F1 score, can improve over YOLOv5. The new architecture upgrades allow for continuously maintaining improvements in accuracy while still being very responsive. In comparison with YOLOv5's already strong foundation, YOLOv8 provides most practical balance for modern ASL translation systems.

SHuBERT surpasses itself in continuous sign language recognition by capturing the lingual details using a transformer architecture, while YOLOv5 is purely detection. It understands full signed sentences with correct grammar but had much greater computational demands than YOLOv5. SHuBERT gives translation quality emphasis while YOLOv5 is more real-time performance instead.

DeepASL can also work in conjunction with an IR sensor to achieve 94.5% word-level accuracy. This opens up new possibilities in low lighting environments where YOLOv5 could deliver. However, the hardware required is special and, therefore, has limited application for general use like YOLOv5. DeepASL performs splendidly in a controlled environment, which YOLOv5 cannot do outside.

Sign Language Transformers decipher complex spatial-temporal relations in signed languages into highly sophisticated attention mechanization to create much deeper linguistic comprehension than YOLOv5 could offer. They have also been reported to achieve state-of-the-art results, but under extremely heavy computational costs, at least in comparison to YOLOv5; for example, YOLOv5 speed and accessibility versus the models' translation quality.

OpenHands which works on pose-based recognition for many sign languages. From that perspective, it examines skeletal structure rather than YOLOv5's pixel-based perspective for detection. So you could have a more generalizable system that can adapt to different signing styles and environments, because it does not focus on just one type of sensor or camera or anything like that. It might not be as computationally efficient as YOLOv5, but it's much more cross-language compatible in that respect.

CONCLUSION:

The model trained here provides a very good and accurate result for the very few signs trained and works in different visible lighting conditions. But the shortcoming is that it is a very few signs that have been trained and this is done because of the training time and data collection constraints. As the data collection was done by me and took a lot of time to upload and label each photo it requires a lot of man hours for a complete ASL translation model.

As a compromise, the model is very accurate and can detect the signs better than what is recommended by other models in their model data.

You would see an increase in performance over YOLOv5's strengths when considering YOLOv8, while SHuBERT and Sign Language Transformers forfeit YOLOv5's speed for linguistic depth. DeepASL is altogether on a different sensor technology, while OpenHands sacrifices efficient detection in the YOLOv5 manner for cross-language adaptability. The best choice depends on what application would value speed over accuracy, hardware constraints, or linguistic complexity.

FUTURE WORK:

This model, though good at what it was set out to do, admittedly has some shortcomings that can be aimed to fix in another iteration or with further work. Namely;

- **Expand the number of signs:** This model covers a very small number of signs that are in ASL. Further work can be done to expand to all the signs in ASL
- **Add More Classes:** Do more thorough testing and make sure with all skin tones and ethnicities to avoid discrimination.
- **Expand Dataset:** Add more signs by taking more images of different hand models of differing skin tones and hand sizes to make sure it works with all types of hands. This involves labelling all the images.
- **Better Augmentations:** The data augmentations done here are very minimal and not very effective. More harsher flipping, mirroring and rotations can be used to make a more robust model.
- **Upgrading to a Better Model:** Maybe moving to a YOLOv8 based model would yield better accuracies. This needs to be tested and implemented.

REFRENCES:

1. https://github.com/ultralytics/yolov5 - YOLOv5 Repository
2. https://pytorch.org/docs/stable/index.html - PyTorch Documentation
3. https://docs.roboflow.com/ - RoboFlow Documentation
4. https://www.youtube.com/watch?v=-UXYAAqm9fE
5. https://github.com/entbappy/Sign-Language-Detection-Project-Yolov5