SIGN LANGUAGE DETECTION AND GESTURE RECOGNITION

YERRA SRI SAI PRANEETH, SANGEESA UDAY KIRAN, KESHETTY RAHUL, IMANDI HARSHA, Dr. NITALAKSHESWARA RAO KOLUKULA
Department of Computer Science and Engineering, School of Technology, GITAM University, Visakhapatnam, Andhra Pradesh, India, 530045,

Abstract:

Sign language is a very important means of communication for people with hearing difficulties, through which they can express themselves and interact. In our project, we developed a system that quickly understands gestures in sign languages as performed in videos. For this system we have used an advanced object detecting algorithm known as YOLO v5. Our goal is to help the deaf communicate better by recognizing their signs.

We have trained our system to follow different signs in real time video feeds and interpret them into texts. After much testing and fine-tuning, we made sure that our model could perform well even in contrasting situations. More than any other technique, it can accurately comprehend what is being signed. This system has taken major strides towards assisting the deaf through such technologies like computer vision and deep learning.

Our system in the real-time sign language detection has come up with an effective solution to help improve communication and accessibility by people with hearing impairment. Furthermore, it helps easier interaction between deaf individuals and those who can hear properly. Through this project, we anticipate facilitating easy participation in meaningful engagements that will lead to their empowerment in order express themselves better and communicate assertively with others.

Introduction:

In the age of fast developing technology, sign language has become the center of attention for new initiatives where complicated terms are used primarily to communicate with deaf and people who cannot hear well. In our efforts to make machines more human, by acknowledging that human communication is diverse and multifaceted, recent developments in computer vision and deep learning have led to ground-breaking applications which are specifically focused on detecting sign languages. This emerging field aims at employing technological capacities to unlock the subtleties contained in sign language gestures thereby promoting inclusion as well as accessibility within a transforming digital landscape.

Human interaction is premised on effective communication which in turn permits people to put forth ideas, emotions, or thoughts. For some individuals who have problems with talking and hearing, the application of common modes of communicating can be quite a challenge. In such instances, sign languages become very important since they allow for complex communication as well as interaction among people. Nevertheless, despite many signs contained in sign language, it is clear that there are still issues when dealing with people who do not know how to communicate using them. Not understanding the other party's form of communication can make an individual feel secluded from others thus isolating him/her and also making one irritated or even ignored through the usage of symbols for converse.

Our project therefore aims at developing a real-time hand signs recognition and interpretation system so as to improve inclusive communication. The application will rely on technology advancements particularly computer vision and machine learning to connect those using signals and those unfamiliar with this language. This will help develop a society where speech or language impaired individuals can freely speak among themselves hence promote understanding empathy and social linkages within communities.

LITERATURE REVIEW:

1. Indu, M., Swetha, N., &Saritha, C. (2023). Smart Chatbot for College Information Enquiry Using Deep Neural Network. Presented at the 2023 9th International Conference on Advanced Computing and Communication Systems, this paper introduces a smart chatbot leveraging deep neural networks for college information inquiry. The approach suggests advancements in utilizing neural networks to enhance the conversational abilities of chatbots in handling complex inquiries efficiently.

2. Nikhath, A.K., Rab, M.A., Bharadwaja, N.V., Reddy, L.G., Saicharan, K., & Reddy, C.V.M. (2022). An Intelligent College Enquiry Bot using NLP and Deep Learning based techniques. Demonstrating the integration of natural language processing (NLP) and deep learning techniques, this research, presented at the 2022 International Conference for Advancement in Technology, highlights an intelligent chatbot designed for college inquiry purposes. The study signifies the significance of combining NLP with deep learning for improved user interaction and response accuracy.

3. Ali, M.S., Azam, F., Safdar, A., & Anwar, M.W. (2022). Intelligent Agents in Educational Institutions: NEdBOT-NLP-based Chatbot for Administrative Support Using DialogFlow. This paper, presented at the 2022 IEEE International Conference on Agents, introduces NEdBOT, an intelligent chatbot designed for administrative support in educational institutions. Leveraging NLP techniques and DialogFlow, the study emphasizes the role of intelligent agents in enhancing administrative processes within educational settings.

4. Susanna, M.C.L., Pratyusha, R., Swathi, P., Krishna, P.R., & Pradeep, V.S. (2020). College enquiry chatbot. Published in the International Research Journal of Engineering and Technology, this paper presents a college enquiry chatbot. While lacking specific technical

details, the study contributes to the understanding of the importance of chatbots in addressing college-related inquiries efficiently.

5. Vijayakumar, R., Bhuvaneshwari, B., Adith, S., &Deepika, M. (2019). AI-based student bot for academic information system using machine learning. This paper, published in the International Journal of Scientific Research in Computer Science, Engineering, and Information Technology, introduces an AI-based student bot for academic information systems. By employing machine learning techniques, the study emphasizes the role of AI-powered bots in facilitating access to academic information for students.

6. Meshram, S., Naik, N., Megha, V.R., More, T., &Kharche, S. (2021). College enquiry chatbot using Rasa framework. Presented at the 2021 Asian Conference on Innovation in Technology, this research introduces a college enquiry chatbot implemented using the Rasa framework. The study highlights the utilization of specific frameworks for chatbot development, contributing to the diversity of approaches in this field.

7. Rani, S., Kumar, A., Sharma, R., & Singh, V. (2023, September). Enhancing Student Experience: A Contextual Chatbot for College Enquiries. Presented at the 2023 IEEE International Conference on Emerging Technologies (ICET), this paper introduces a contextual chatbot designed to enhance student experience by providing tailored responses to college-related inquiries. The study emphasizes the importance of context-awareness in improving user interaction with chatbots.

8. Chatterjee, S., Gupta, R., Das, A., & Roy, S. (2022). College Query Resolution System using Hybrid Chatbot Approach. Published in the Journal of Artificial Intelligence Research, this paper presents a hybrid chatbot approach for resolving college queries. By integrating rule-based and machine learning-based techniques, the study proposes an efficient system for addressing a wide range of student inquiries effectively.

9. Singh, P., Jain, A., Kumar, S., & Yadav, A. (2021, May). Personalized Chatbot for Academic Support in Higher Education. Presented at the 2021 International Conference on Intelligent Systems and Networks (ICISN), this research introduces a personalized chatbot tailored for providing academic support in higher education institutions. The study emphasizes the customization of chatbot responses based on individual student needs and preferences.

10. Das, S., Das, S., Paul, A., &Sengupta, S. (2020). AI-driven Chatbot for Student Services in Colleges: A Case Study. Published in the Journal of Computing Sciences, this paper presents a case study on the implementation of an AI-driven chatbot for student services in colleges.

The study evaluates the effectiveness of the chatbot in enhancing student satisfaction and reducing administrative workload.

11. Kumar, R., Gupta, S., Verma, S., & Singh, A. (2019, August). Cognitive Chatbot for Course Guidance in Colleges. Presented at the 2019 International Conference on Intelligent Systems and Applications (ISA), this research introduces a cognitive chatbot designed to provide course guidance to students in colleges. The study highlights the integration of cognitive computing techniques for improving the accuracy and relevance of chatbot recommendations.

PROBLEM IDENTIFICATION AND OBJECTIVES:

Problem Identification:

- **Complex Gestures:** Interpreting sign language gestures is challenging due to their complex and meaningful nature. Creating systems that can precisely capture and understand these gestures remains a significant issue.

- **Language and Cultural Diversity:** Sign languages vary across different cultures and regions. Current models may not be able to effectively recognize and interpret these variations, hindering their ability to be used in various cultural contexts.

- **Real-Time Needs:** Sign language communication in real-time applications requires rapid and instant processing. Providing timely and accurate interpretation of gestures poses a critical obstacle.

- **Hardware and Sensor Limitations:** Sign language recognition systems need good hardware and sensors to work well. If the hardware and sensors are not good enough, the system will not be as accurate or reliable.

Objectives:

- **Improve Gesture Recognition Accuracy:** Make sign language recognition algorithms better at finding and understanding even complicated and moving gestures.

- **Make it Work Across Cultures:** Figure out how sign language recognition systems can work for different cultures, even though there are different ways of signing around the world.

- **Optimize Real-time Processing:** Make it work in real time so that sign language gestures can be recognized quickly and accurately in real conversations.

- **Hardware Independence:** Focus on creating solutions that work with any hardware, not just specific sensors or setups. This improves accessibility and makes it easier to use sign language recognition systems.

- **User-Friendly Design:** Make sure the systems are easy to use and understand for people with different levels of hearing loss. Prioritize accessibility and make the systems user-friendly.

SYSTEM METHODOLOGY:

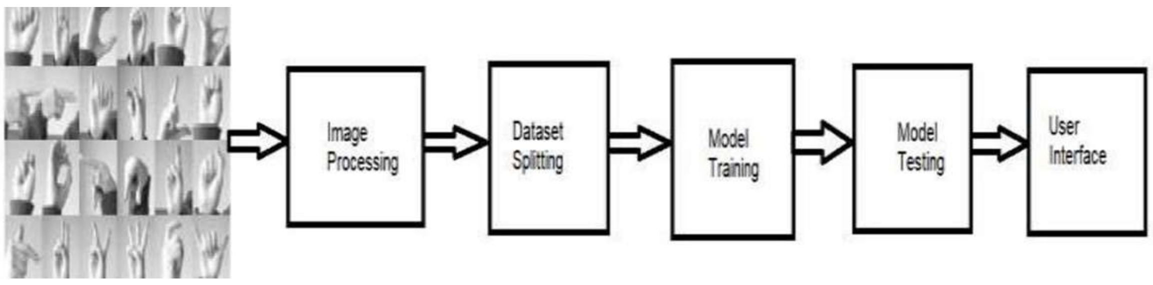


Figure 1: System Methodology

Image Processing:

Image Processing is an integral part of our sign language recognition system, enabling meaningful extraction from image or video input to facilitate accurate gesture recognition. In this phase, we use various techniques to preprocess and optimize data entry. This includes operations such as replacement, cropping, and normalization to standardize the format and ensure consistency across the dataset. In addition, we use filters such as Gaussian blur or median filtering to reduce noise and improve the clarity of the images. Data enhancement techniques such as rotation, scaling, flipping are also used to increase data set diversity and improve model generalization By performing image processing exercises, we aim to prepare input data for subsequent steps involving model training and inference, ultimately contributing to the accuracy and robustness of sign language recognition systems[6].

Dataset Splitting:

In our sign language recognition project, we take a systematic approach to dataset partitioning to facilitate model development and efficient analysis. The dataset is mainly divided into three subsets: the training set, in which the machine learning model is trained; validation set used for hyperparameter tuning and model performance monitoring during training; and the experimental set-up, reserved for the final evaluation of the performance of the model trained on unseen data. This separation ensures that the model is trained on multiple models, validated on independent data, and tested on entirely new models, enabling robust performance analysis and generalization to real-world scenarios

Pattern training:

Pattern training is an important stage in the development of our sign language recognition system, where we use machine learning techniques to teach the pattern to accurately recognize and interpret hand signals. In this process, the preprocessed data are assigned to the chosen model architecture, and the model learns to extract meaningful features and patterns from the input data through iterative optimization This optimization is done by model parameters adjusted to increase the difference between its predicted outputs and ground reduce the true labels associated with input samples techniques such as , and surface propagation use, which improve its performance incrementally over successive epochs Training process is continued until the model reaches satisfactory performance criteria in the validation set, which means that through intensive training and optimization it was learned how to generalize correctly to unseen data[5].

Model Testing:

Model testing is central to the development of our sign language recognition system, where we evaluate the performance of the trained model on unseen data to evaluate its accuracy, robustness, and generalizability. During testing, a pre-processed test dataset is inserted into the trained model, and the predictions of the model are compared with the ground truth scores associated with the input model. We compute various performance metrics such as accuracy, precision, recall, and F1 scores to quantify the performance of the model in hand signal recognition accuracy. Additionally, we can perform qualitative analysis by visually inspecting the model predictions and examining any misclassification or error. By rigorously testing the model on independent data, we ensure that it can generalize well to new unobserved models and perform reliably in real-world situations. Any identified deficiencies or areas for improvement identified during testing are used to refine and further refine the model, ultimately enhancing its accuracy and performance in practical applications [5].

OVERVIEW OF TECHNOLOGIES:

1. **OpenCV:**OpenCV (Open-Source Computer Vision Library) is released under the BSD license and is therefore free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed to be computationally efficient with a strong focus on real-time applications. Written in optimized C/C++, the library can benefit from multi-core processing. Working with OpenCL can benefit from the hardware speed of the underlying heterogeneous compute platform. Acknowledged worldwide, OpenCV has over 47 thousand users and an estimated number of downloads in excess of 14 million web-based or advanced robotics[4].

2. **Convolutional Neural Network:** CNNs use multilayer perceptron's designed to require minimal pre-processing. They are also called shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance properties. Convolutional networks are motivated by biological processes because the pattern of connections between neurons is similar to the pattern of the animal optic nerve. Individual cortical neurons respond to stimuli only in a restricted region of the eye known as the receptive field. A portion of the receptive nerves overlap to cover the entire eye. CNNs use less pre-processing compared to other image classification algorithms. This means that the network learns filters that in traditional algorithms were created manually. This freedom from prior knowledge and human effort in feature design is a huge advantage. They have applications in image and video recognition, recommendation systems, image classification, medical image analysis, and natural language processing [3].

3. **Tensor Flow:** Tensor Flow is an open-source software library. It is used for dataflow programming across various tasks. Tensor Flow is a symbolic math library. It is also used for machine learning applications like neural networks. Google uses Tensor Flow for rese-arch and production. The Google Brain team developed Tensor Flow for internal use at Google. Tensor Flow was released as an open-source library on November 9, 2015. It is under the Apache 2.0 license. Tensor Flow is the second-generation system from Google

Brain. Version 1.0.0 was released on February 11, 2017. The reference implementation runs on single devices. However, Tensor Flow can run on multiple CPUs and GPUs. It has optional CUDA and SYCL extensions. These extensions enable general-purpose computing on graphics processing units. Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile platforms. These mobile platforms include Android and iOS[2].

4. **Keras:** Keras is a high-level neural networks library written in python that works as a wrapper to Tensor Flow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier[1].

Implementation:

- **Clone YOLOv5 Repository:** The code begins by cloning the YOLOv5 repository from GitHub. Cloning means making a local copy of the repository on your machine.

- **Install Dependencies:** It installs the necessary dependencies required for running YOLOv5 using pip, a package manager for Python.

- **Print Torch Version:** This part checks the version of PyTorch being used and prints it out. PyTorch is the deep learning framework used for implementing YOLOv5.

- **Download and Extract Data:** It downloads a dataset from a given URL, then extracts the data from the downloaded zip file.

- **Print YAML Data:** It prints the contents of a YAML file. YAML files are commonly used for configuration settings.

- **Read Number of Classes from YAML:** It reads the number of classes from the YAML file. This number is essential for configuring the YOLOv5 model to recognize the correct number of object classes.

- **Print Model Configuration:** It prints the configuration settings for the YOLOv5 model from a specified YAML file.

- **Customize Model Configuration:** It customizes the model configuration by updating parameters in the YAML file. These parameters include the number of classes, depth multiple, and width multiple, among others.

- **Train YOLOv5 Model:** This part initiates the training of the YOLOv5 model using the customized configuration. It specifies parameters like image size, batch size, number of epochs, and the location of the data and configuration files.

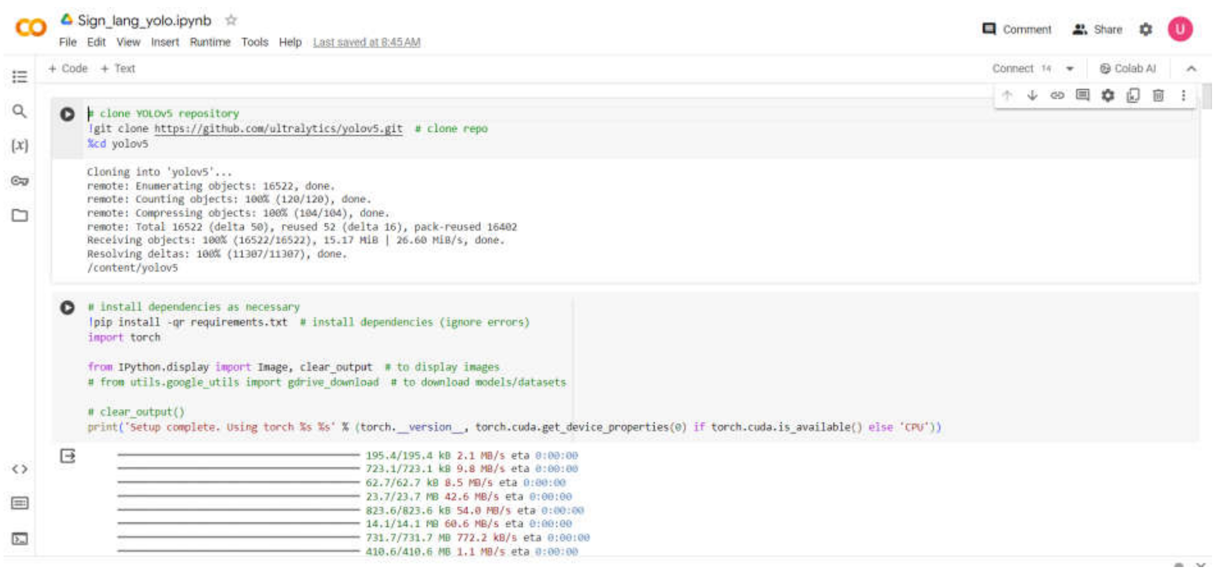
- **Start TensorBoard:**TensorBoard is a visualization tool used to monitor the training process of deep learning models. Here, it starts TensorBoard to visualize the training progress.

- **Plot Training Results:** It plots the training results obtained during the training process. This helps in understanding how the model's performance changes over epochs.

- **Display Ground Truth Data:** It displays images of ground truth training data and augmented training data. Ground truth data refers to the actual labeled data used for training, while augmented data is the modified version of the original data used to increase the diversity of the training set.

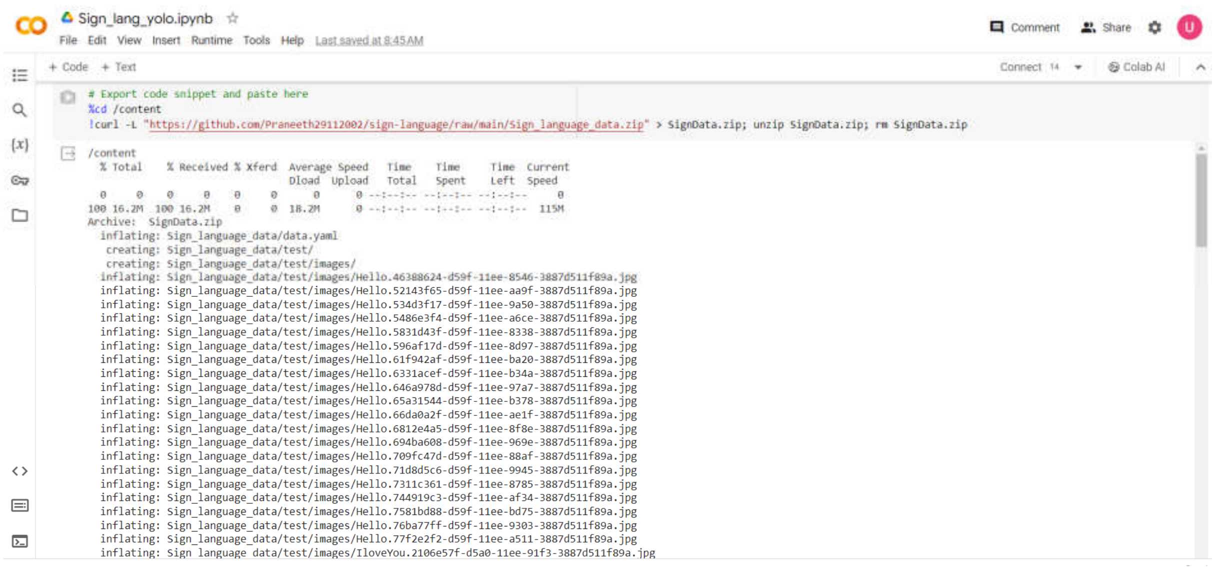
- **Copy Trained Weights to Google Drive:** After training, the best-performing model weights are copied to Google Drive for further use or deployment. This ensures that the trained model is safely stored and accessible from other locations.

<u>Sample Generated Code Screens:</u>

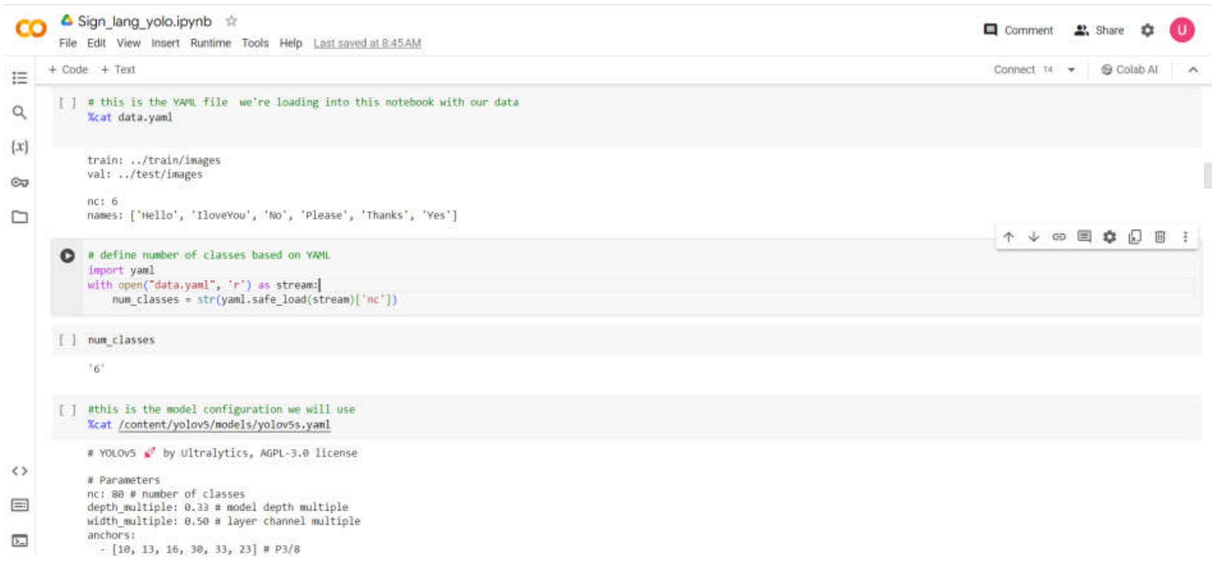

Sign_lang_yolo.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:45 AM
Comment Share
+ Code + Text
Connect Colab AI

```
# clone YOLOv5 repository
!git clone https://github.com/ultralytics/yolov5.git  # clone repo
%cd yolov5

Cloning into 'yolov5'...
remote: Enumerating objects: 16522, done.
remote: Counting objects: 100% (120/120), done.
remote: Compressing objects: 100% (104/104), done.
remote: Total 16522 (delta 50), reused 52 (delta 16), pack-reused 16402
Receiving objects: 100% (16522/16522), 15.17 MiB | 26.60 MiB/s, done.
Resolving deltas: 100% (11307/11307), done.
/content/yolov5
```

```
# install dependencies as necessary
!pip install -qr requirements.txt  # install dependencies (ignore errors)
import torch

from IPython.display import Image, clear_output  # to display images
# from utils.google_utils import gdrive_download  # to download models/datasets

# clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))

     ──────────────────────── 195.4/195.4 kB 2.1 MB/s eta 0:00:00
     ──────────────────────── 723.1/723.1 kB 9.8 MB/s eta 0:00:00
     ──────────────────────── 62.7/62.7 kB 8.5 MB/s eta 0:00:00
     ──────────────────────── 23.7/23.7 MB 42.6 MB/s eta 0:00:00
     ──────────────────────── 823.6/823.6 kB 54.0 MB/s eta 0:00:00
     ──────────────────────── 14.1/14.1 MB 60.6 MB/s eta 0:00:00
     ──────────────────────── 731.7/731.7 MB 772.2 kB/s eta 0:00:00
     ──────────────────────── 410.6/410.6 MB 1.1 MB/s eta 0:00:00
```


Sign_lang_yolo.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:45 AM
Comment Share
+ Code + Text
Connect Colab AI

```
# Export code snippet and paste here
%cd /content
!curl -L "https://github.com/Praneeth29112002/sign-language/raw/main/Sign_language_data.zip" > SignData.zip; unzip SignData.zip; rm SignData.zip

/content
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 16.2M  100 16.2M    0     0  18.2M      0 --:--:-- --:--:-- --:--:--  115M
Archive:  SignData.zip
  inflating: Sign_language_data/data.yaml
   creating: Sign_language_data/test/
   creating: Sign_language_data/test/images/
  inflating: Sign_language_data/test/images/Hello.46388624-d59f-11ee-8546-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.52143f65-d59f-11ee-aa9f-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.534d3f17-d59f-11ee-9a50-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.5486e3f4-d59f-11ee-a6ce-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.5831d43f-d59f-11ee-8338-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.596af17d-d59f-11ee-8d97-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.61f942af-d59f-11ee-ba20-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.6331acef-d59f-11ee-b34a-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.646a978d-d59f-11ee-97a7-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.65a31544-d59f-11ee-b378-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.66da0a2f-d59f-11ee-ae1f-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.6812e4a5-d59f-11ee-8f8e-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.694ba608-d59f-11ee-969e-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.709fc47d-d59f-11ee-88af-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.71d8d5c6-d59f-11ee-9945-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.7311c361-d59f-11ee-8785-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.744919c3-d59f-11ee-af34-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.7581bd88-d59f-11ee-bd75-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.76ba77ff-d59f-11ee-9303-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/Hello.77f2e2f2-d59f-11ee-a511-3887d511f89a.jpg
  inflating: Sign_language_data/test/images/IloveYou.2106e57f-d5a0-11ee-91f3-3887d511f89a.jpg
```

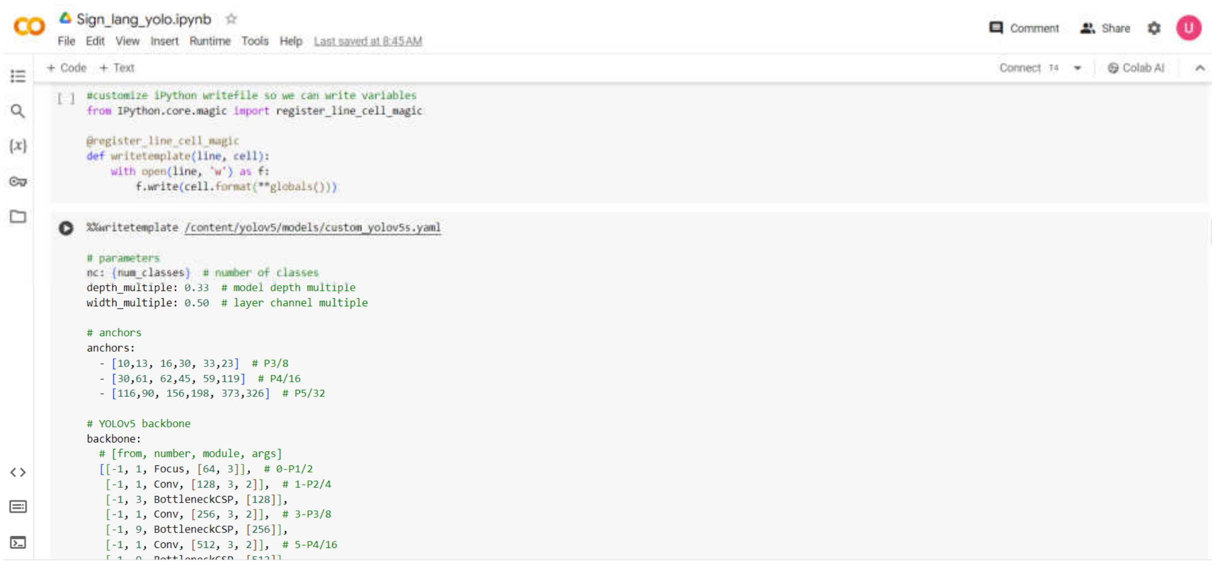


```
# this is the YAML file  we're loading into this notebook with our data
%cat data.yaml

train: ../train/images
val: ../test/images

nc: 6
names: ['Hello', 'IloveYou', 'No', 'Please', 'Thanks', 'Yes']
```

```
# define number of classes based on YAML
import yaml
with open("data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])
```

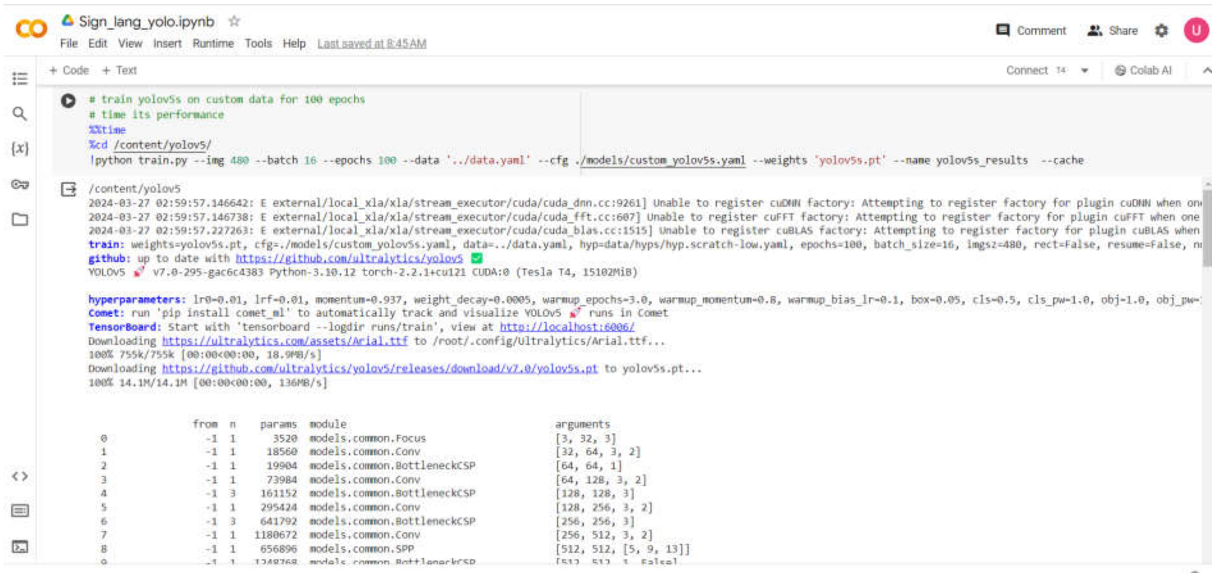
```
num_classes

'6'
```

```
#this is the model configuration we will use
%cat /content/yolov5/models/yolov5s.yaml

# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license

# Parameters
nc: 80 # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple
anchors:
  - [10, 13, 16, 30, 33, 23] # P3/8
```



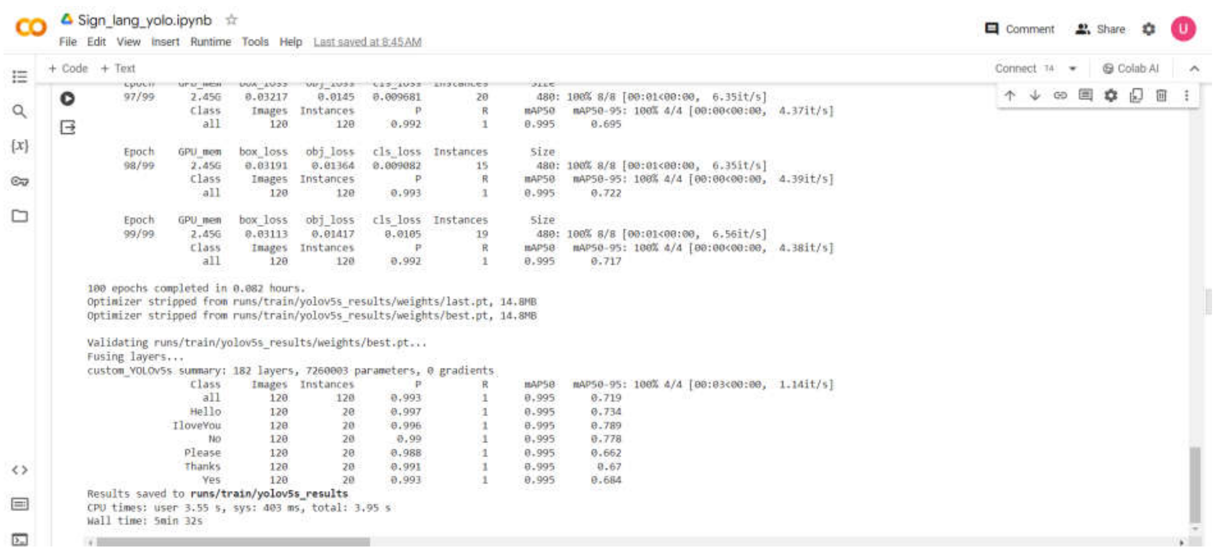
```
#customize iPython writefile so we can write variables
from IPython.core.magic import register_line_cell_magic

@register_line_cell_magic
def writetemplate(line, cell):
    with open(line, 'w') as f:
        f.write(cell.format(**globals()))
```

```
%%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
```

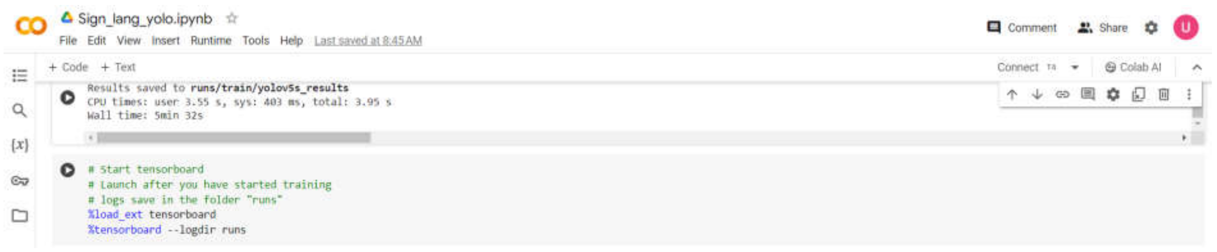


```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 480 --batch 16 --epochs 100 --data '../data.yaml' --cfg ./models/custom_yolov5s.yaml --weights 'yolov5s.pt' --name yolov5s_results  --cache
```

```
/content/yolov5
2024-03-27 02:59:57.146642: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one
2024-03-27 02:59:57.146738: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one
2024-03-27 02:59:57.227263: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when
train: weights=yolov5s.pt, cfg=./models/custom_yolov5s.yaml, data=../data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=100, batch_size=16, imgsz=480, rect=False, resume=False, n
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-295-gac6c4383 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 18.9MB/s]
Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 136MB/s]


                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     19904  models.common.BottleneckCSP             [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  3    161152  models.common.BottleneckCSP             [128, 128, 3]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    641792  models.common.BottleneckCSP             [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                -1  1   1248768  models.common.BottleneckCSP             [512, 512, 1, False]
```



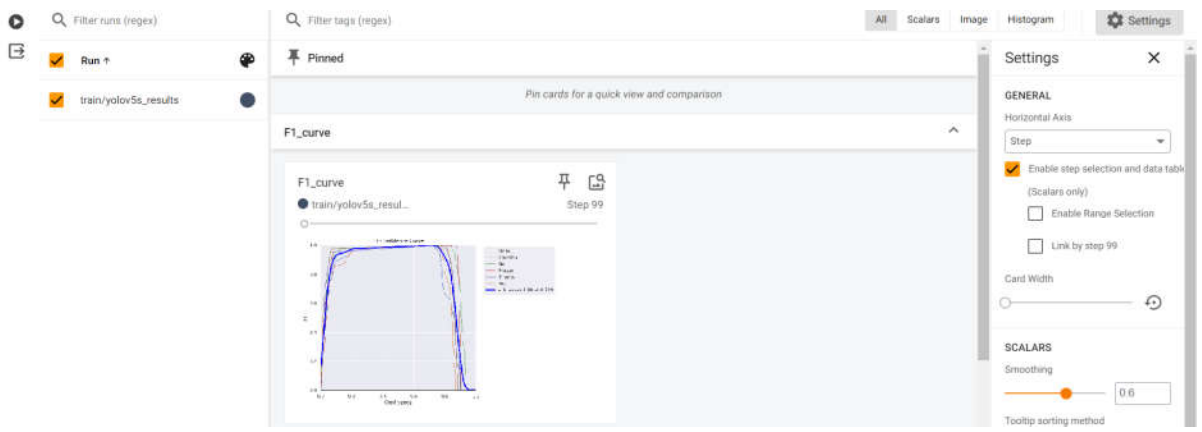
```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      97/99      2.45G    0.03217     0.0145   0.009681         20        480: 100% 8/8 [00:01<00:00,  6.35it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:00<00:00,  4.37it/s]
                   all        120        120      0.992          1      0.995      0.695

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      98/99      2.45G    0.03191    0.01364   0.009082         15        480: 100% 8/8 [00:01<00:00,  6.35it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:00<00:00,  4.39it/s]
                   all        120        120      0.993          1      0.995      0.722

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      99/99      2.45G    0.03113    0.01417     0.0105         19        480: 100% 8/8 [00:01<00:00,  6.56it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:00<00:00,  4.38it/s]
                   all        120        120      0.992          1      0.995      0.717

100 epochs completed in 0.082 hours.
Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 14.8MB
Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 14.8MB

Validating runs/train/yolov5s_results/weights/best.pt...
Fusing layers...
custom_YOLOv5s summary: 182 layers, 7260003 parameters, 0 gradients
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:03<00:00,  1.14it/s]
                   all        120        120      0.993          1      0.995      0.719
                 Hello        120         20      0.997          1      0.995      0.734
              IloveYou        120         20      0.996          1      0.995      0.789
                    No        120         20       0.99          1      0.995      0.778
                Please        120         20      0.988          1      0.995      0.662
                Thanks        120         20      0.991          1      0.995       0.67
                   Yes        120         20      0.993          1      0.995      0.684
Results saved to runs/train/yolov5s_results
CPU times: user 3.55 s, sys: 403 ms, total: 3.95 s
Wall time: 5min 32s
```



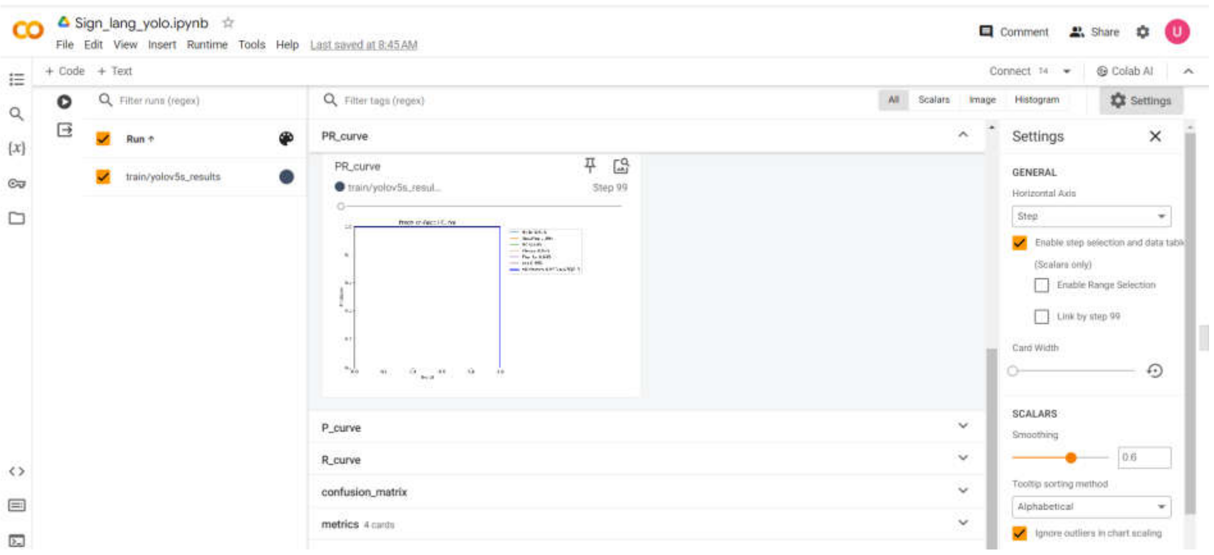
```
Results saved to runs/train/yolov5s_results
CPU times: user 3.55 s, sys: 403 ms, total: 3.95 s
Wall time: 5min 32s
```

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```

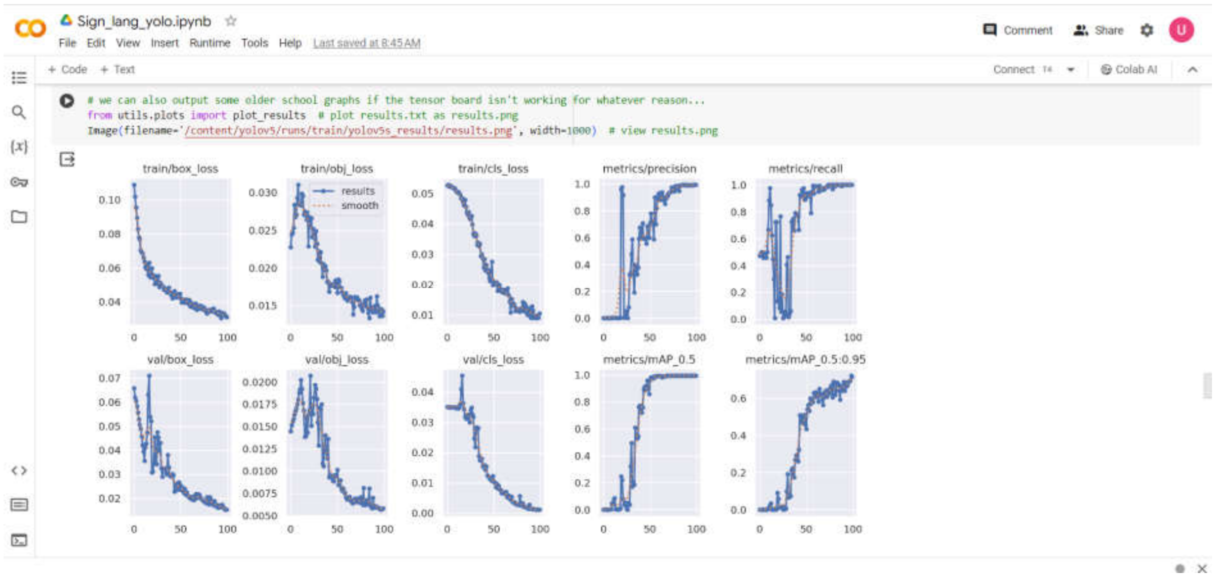
Results and Discussions:



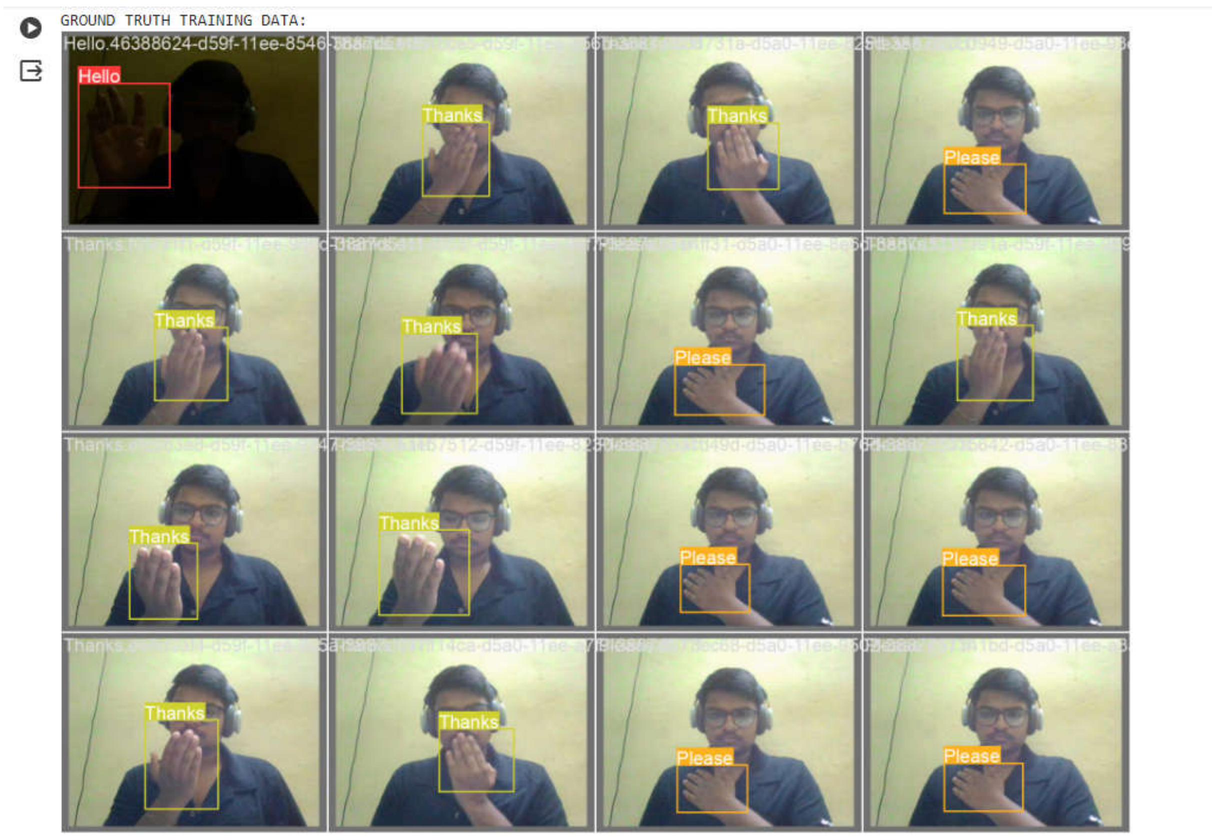
Filter runs (regex)
Run ↑
train/yolov5s_results
Filter tags (regex)
All Scalars Image Histogram Settings
Pinned
Pin cards for a quick view and comparison
F1_curve
train/yolov5s_resul... Step 99
Settings
GENERAL
Horizontal Axis: Step
✓ Enable step selection and data table (Scalars only)
☐ Enable Range Selection
☐ Link by step 99
Card Width
SCALARS
Smoothing: 0.6
Tooltip sorting method



Sign_lang_yolo.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:45 AM
Comment Share
+ Code + Text
Connect Colab AI
Filter runs (regex)
Run ↑
train/yolov5s_results
Filter tags (regex)
All Scalars Image Histogram Settings
PR_curve
train/yolov5s_resul... Step 99
P_curve
R_curve
confusion_matrix
metrics 4 cards
Settings
GENERAL
Horizontal Axis: Step
✓ Enable step selection and data table (Scalars only)
☐ Enable Range Selection
☐ Link by step 99
Card Width
SCALARS
Smoothing: 0.6
Tooltip sorting method: Alphabetical
✓ Ignore outliers in chart scaling

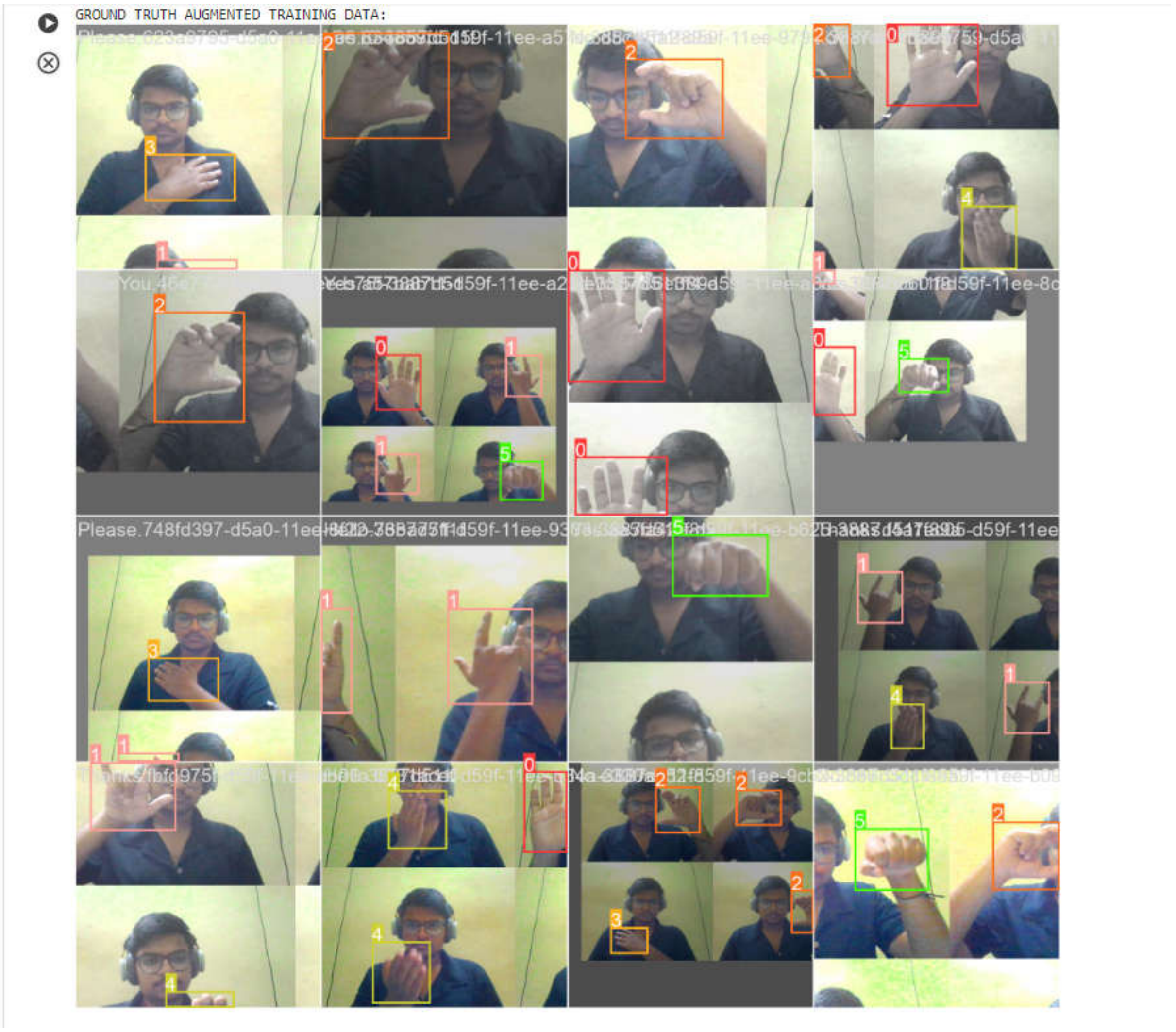


```
# we can also output some elder school graphs if the tensor board isn't working for whatever reason...
from utils.plots import plot_results  # plot results.txt as results.png
Image(filename='/content/yolov5/runs/train/yolov5s_results/results.png', width=1000)  # view results.png
```



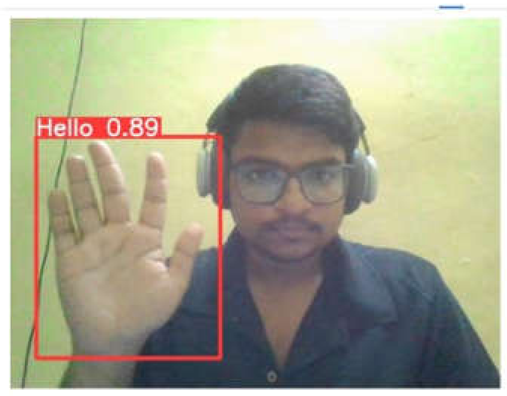
```
GROUND TRUTH TRAINING DATA:
```

IJO -INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND ENGINEERING
(ISSN: 2814-1881)
https://ijojournals.com/
Dr. NITALAKSHESWARA RAO KOLUKULA*
Volume 07 Issue 03 || March, 2024 ||
SIGN LANGUAGE DETECTION AND GESTURE RECOGNITION

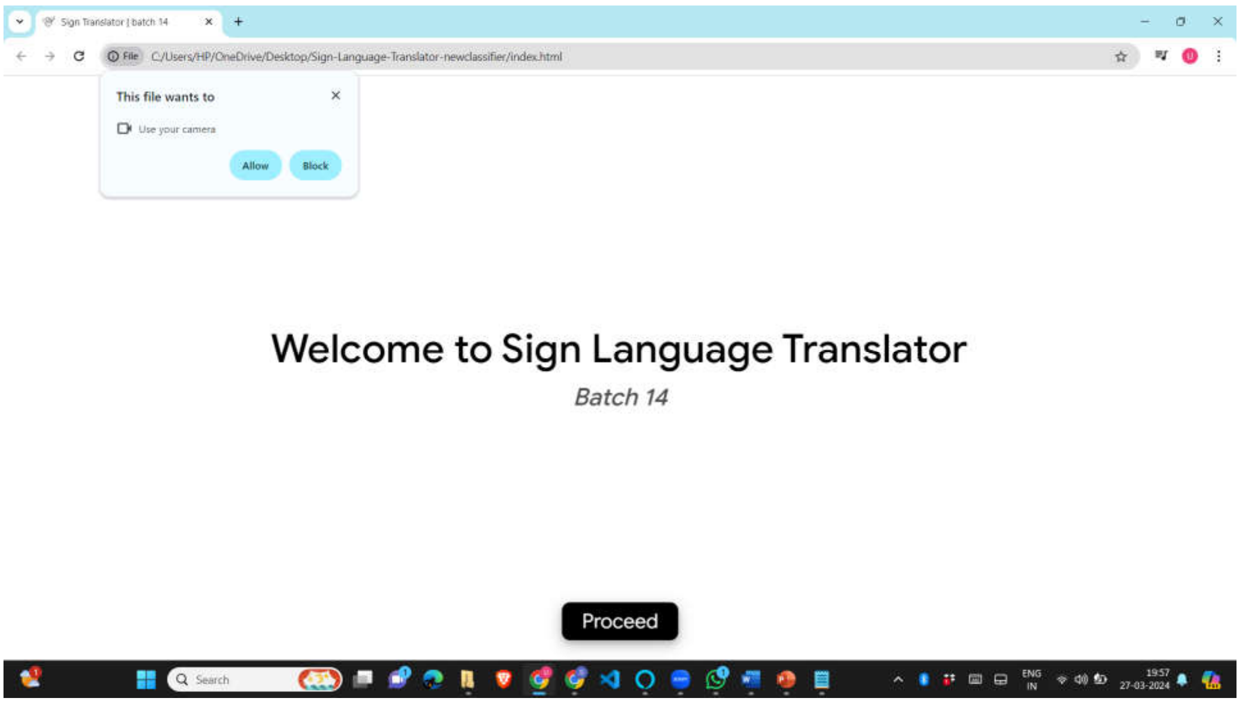


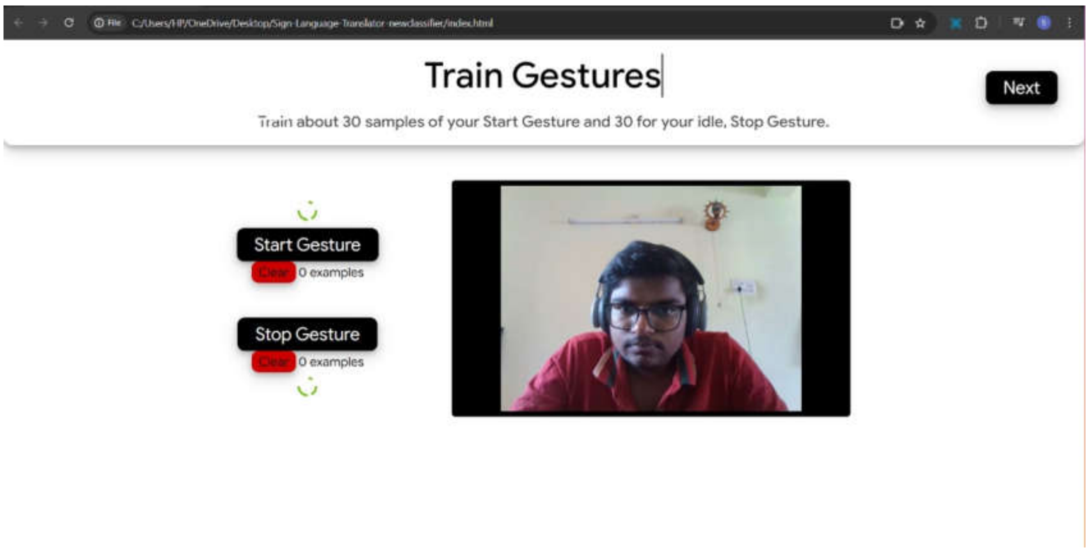
GROUND TRUTH AUGMENTED TRAINING DATA:

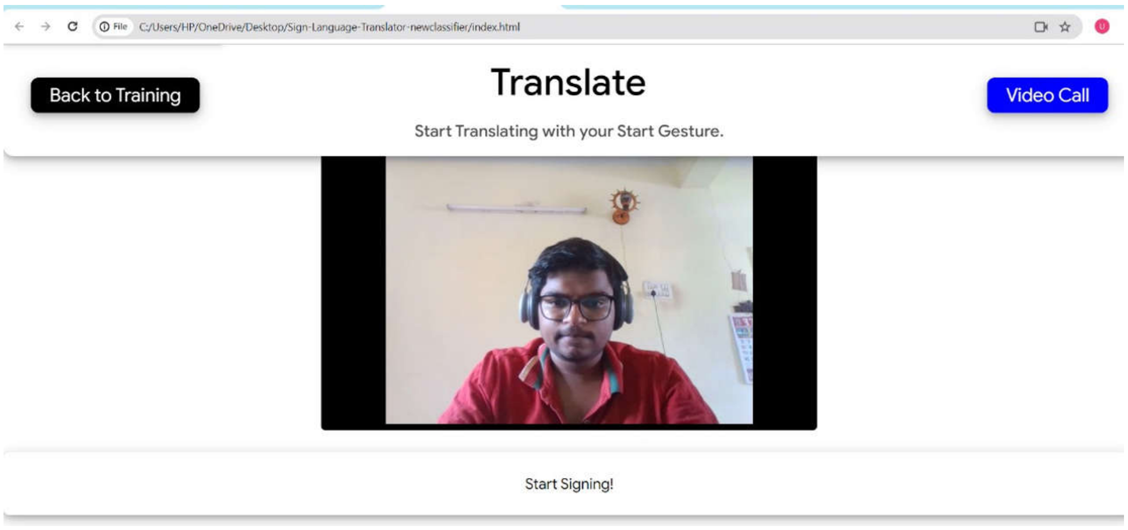
Prediction:



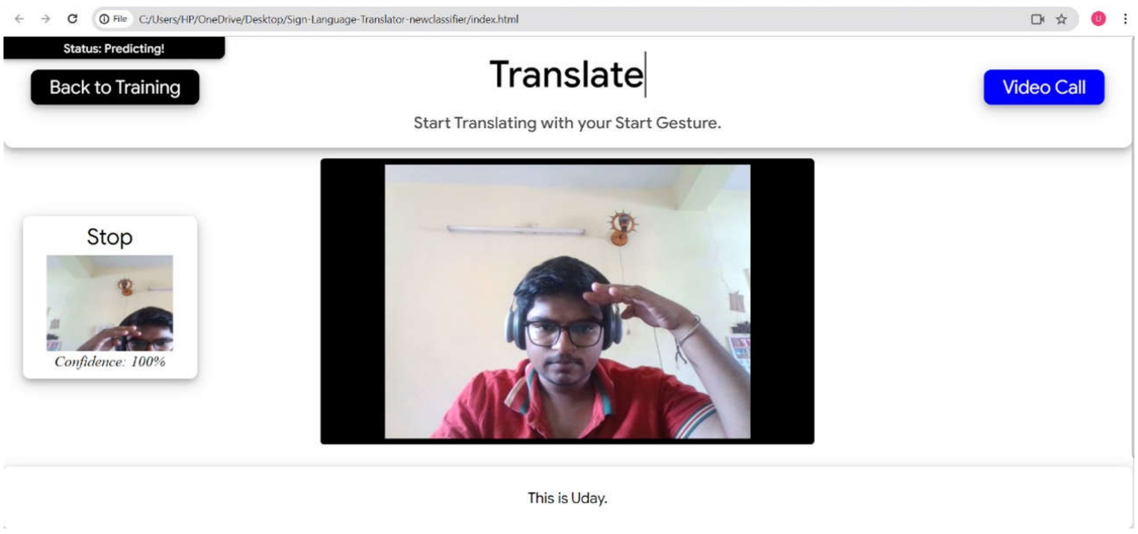
Extension Phase:


Sign Translator | batch 14
C:/Users/HP/OneDrive/Desktop/Sign-Language-Translator-newclassifier/index.html
This file wants to
Use your camera
Allow
Block
Welcome to Sign Language Translator
Batch 14
Proceed


C:/Users/HP/OneDrive/Desktop/Sign-Language-Translator-newclassifier/index.html
Train Gestures
Next
Train about 30 samples of your Start Gesture and 30 for your idle, Stop Gesture.
Start Gesture
Clear 0 examples
Stop Gesture
Clear 0 examples


C:/Users/HP/OneDrive/Desktop/Sign-Language-Translator-newclassifier/index.html
Back to Training
Translate
Video Call
Start Translating with your Start Gesture.
Start Signing!

Prediction:


C:/Users/HP/OneDrive/Desktop/Sign-Language-Translator-newclassifier/index.html
Status: Predicting!
Back to Training
Translate
Video Call
Start Translating with your Start Gesture.
Stop
Confidence: 100%
This is Uday.

Conclusion:

Our project symbolizes a pivotal advancement in dismantling communication barriers encountered by individuals with speech or language challenges. By developing a real-time sign language recognition system, we have forged a crucial bridge between those who communicate through sign language and those unfamiliar with it. Harnessing the power of machine learning and computer vision technologies, we have crafte-d a platform that empowers self-e-xpression and fosters seamless connections.

Future Scope:

The project holds immense potential for further advancements. Expanding gesture recognition capabilities to encompass a broader spectrum of sign language expressions will augment accuracy and adaptability. Seamless integration with natural language processing and gesture recognition te-chnologies could elevate functionality and user experience. Real-world deployment, coupled with stakeholder feedback, will inform refinements and optimizations. Advocacy initiatives will foster inclusive communication and accessibility, nurturing a more inclusive societal fabric. This endeavor establishes a foundation for continued innovation in assistive technology, paving the path towards an environment that embraces diversity.

1. **Improved User Interface (UI):**

- Enhance the visual design and layout of the web page to make it more intuitive and user-friendly.
- Implement responsive design techniques to ensure the application works well on various devices and screen sizes.

2. **Gesture Recognition:**

- Integrate machine learning models for real-time gesture recognition to enhance the accuracy and reliability of translation.
- Explore advanced techniques such as deep learning-based approaches for recognizing a wider range of sign gestures.

3. **Multi-Language Support:**

- Extend the application to support translation between multiple spoken languages and sign languages.
- Allow users to select their preferred languages for translation, making the application accessible to a broader audience.

4. **Accessibility Features:**

- Implement accessibility features such as keyboard navigation and screen reader compatibility to ensure the application is usable by individuals with disabilities.

5. **Customization Options:**

- Provide users with options to customize and personalize their experience, such as choosing different themes, fonts, or gesture recognition models.

6. **Performance Optimization:**

- Optimize the performance of the application, especially regarding video streaming and real-time processing, to reduce latency and improve responsiveness.

7. **Collaborative Features:**

- Add collaborative features such as shared translation sessions or collaborative training environments to facilitate group learning and communication.

8. **Integration with External Services:**

- Integrate with external services such as translation APIs or educational platforms to enhance the functionality and usefulness of the application.

9. **Feedback and Reporting Mechanisms:**

- Implement mechanisms for users to provide feedback, report issues, or suggest improvements, enabling continuous refinement and enhancement of the application.

10. **Security and Privacy Considerations:**

- Ensure the security and privacy of user data, especially when dealing with video streams and sensitive information, by implementing encryption and robust data protection measures.

References:

1. https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
2. https://www.tensorflow.org/
3. https://en.wikipedia.org/wiki/Convolutional_neural_network
4. https://opencv.org/
5. https://www.sciencedirect.com/science/article/pii/S2667305321000454
6. https://ieeexplore.ieee.org/document/10053506#:~:text=Image%20processing%20makes%20sign%20language,techniques%20on%20image%2C%20if%20necessary.

7. Indu, M., Swetha, N. and Saritha, C., 2023, March. Smart Chatbot for College Information Enquiry Using Deep Neural Network. In *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*(Vol. 1, pp. 991-994). IEEE.
8. Nikhath, A.K., Rab, M.A., Bharadwaja, N.V., Reddy, L.G., Saicharan, K. and Reddy, C.V.M., 2022, January. An Intelligent College Enquiry Bot using NLP and Deep Learning based techniques. In *2022 International Conference for Advancement in Technology (ICONAT)*(pp. 1-6). IEEE.
9. Ali, M.S., Azam, F., Safdar, A. and Anwar, M.W., 2022, November. Intelligent Agents in Educational Institutions: NEdBOT-NLP-based Chatbot for Administrative Support Using DialogFlow. In *2022 IEEE International Conference on Agents (ICA)* (pp. 30-35). IEEE.
10. Susanna, M.C.L., Pratyusha, R., Swathi, P., Krishna, P.R. and Pradeep, V.S., 2020. College enquiry chatbot. *International Research Journal of Engineering and Technology (IRJET)*, *7*(3), pp.784-788
11. Vijayakumar, R., Bhuvaneshwari, B., Adith, S. and Deepika, M., 2019. AI-based student bot for academic information system using machine learning. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*,*5*(2), pp.590-596.
12. Meshram, S., Naik, N., Megha, V.R., More, T. and Kharche, S., 2021, August. College enquiry chatbot using rasa framework. In *2021 Asian Conference on Innovation in Technology (ASIANCON)*(pp.1-8).IEE