Configuration Manual

MSc Research Project
Data Analytics

Aji Poovannapoikayil
Student ID: x22184431

School of Computing
National College of Ireland

Supervisor: Dr David Hamill

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Aji Vishwambharan Poovannapoikayil		
Student ID:	x22184431		
Programme:	MSc. Data Analytics	**Year:**	2023-2024
Module:	MSc. Research Project		
Lecturer:	Dr David Hamill		
Submission Due Date:	12/08/2024		
Project Title:	Indian Sign Language Detection and Translation using Deep Learning and Text-to-Speech		
Word Count:	2030	**Page Count:**	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Aji Poovannapoikayil

Date: 12th August, 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Aji Poovannapoikayil
x22184431

1 Introduction

The manual provides detailed instructions on how to set up, configure, and run the Indian Sign Language (ISL) and Translation system using Deep Learning models and a Text-to-speech (TTS) engine. This document covers the entire setup process to replicate results, including hardware and software requirements, installation steps, and system configuration.

2 System Requirements

2.1 Hardware Requirements

The configuration of the machine are as follows:

Table 1: Machine Hardware Configurations

Processor	Minimum Intel i5 or equivalent
RAM	Minimum 8GB (16GB Recommended)
GPU	NVIDIA GPU with CUDA support (Optional but recommended for faster model training locally)
Storage	At least 50GB of disk space

2.2 Software Requirements

To ensure smooth execution of the code, run it on windows 10 or 11 or any other system with similar requirements, using Python version 3.8 or higher.

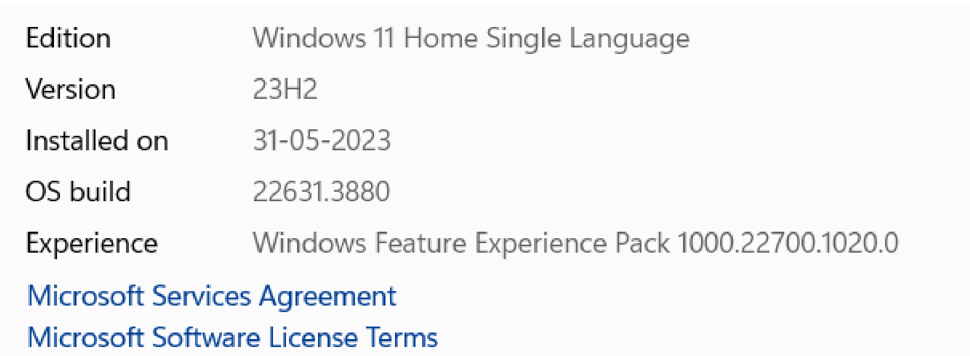

Figure 1: Windows Operating System

The python code was written, maintained, and run using Jupyter notebook from the Anaconda Navigator platform for data pre-processing, training, and other tasks. The Anaconda can be downloaded for windows 64-bit. After installation, launch JupyterLab to run the notebook. Follow the installation steps, and once installed, open Anaconda Navigator and type 'jupyter notebook' in the command line to launch it directly in your browser.

Link to download: https://docs.anaconda.com/anaconda/install/windows/

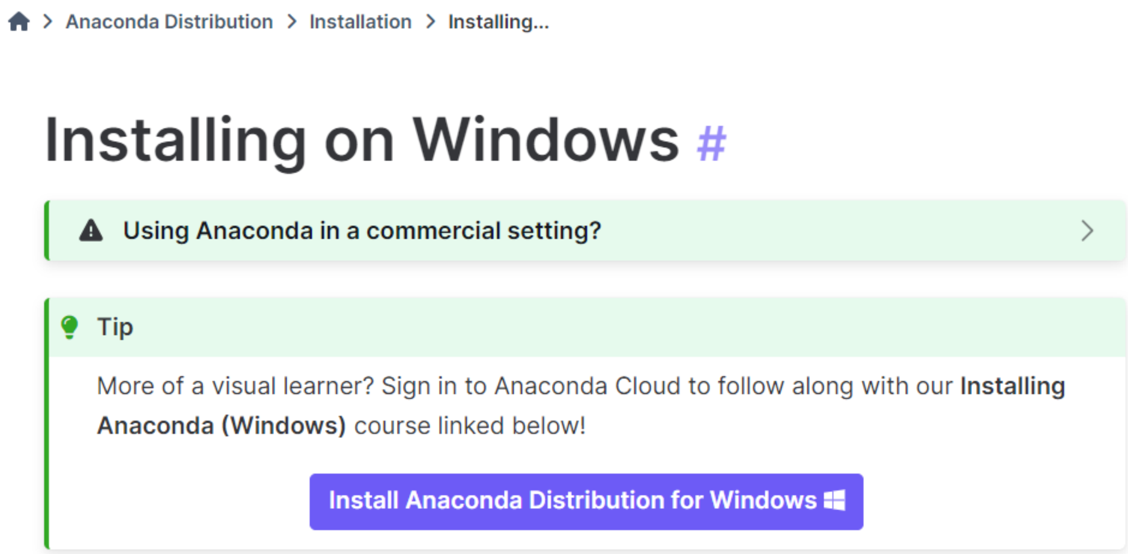

Figure 2: Anaconda Installation

3 Python Libraries Setup

The execution of notebook for data processing, the following python libraries must be installed in the local environment after setting up Jupyter Notebook as mentioned in Section 2.2. The packages are installed using 'pip' command.

Table 2: Python Libraries

Shutil	! pip install shutil
Numpy	! pip install numpy
Pandas	! pip install pandas
Seaborn	! pip install seaborn
Matplotlib	! pip install matplotlib
Torch	! pip install torch
Torchvision	! pip install torchvision
Torchaudio	! pip install torchaudio
Opencv-python	! pip install opencv-python
Opencv-python-headless	! pip install opencv-python-headless
Ultralytics	! pip install ultralytics

'LabelImg' is a graphical Image annotation tool that must be downloaded separately for labelling YOLO datasets. The python library is necessary for preparing the data in YOLO format. After installation, library can be run by opening a command line and executing the Python command 'LabelImg.py'. This will launch the tool, as shown in Figure 3.

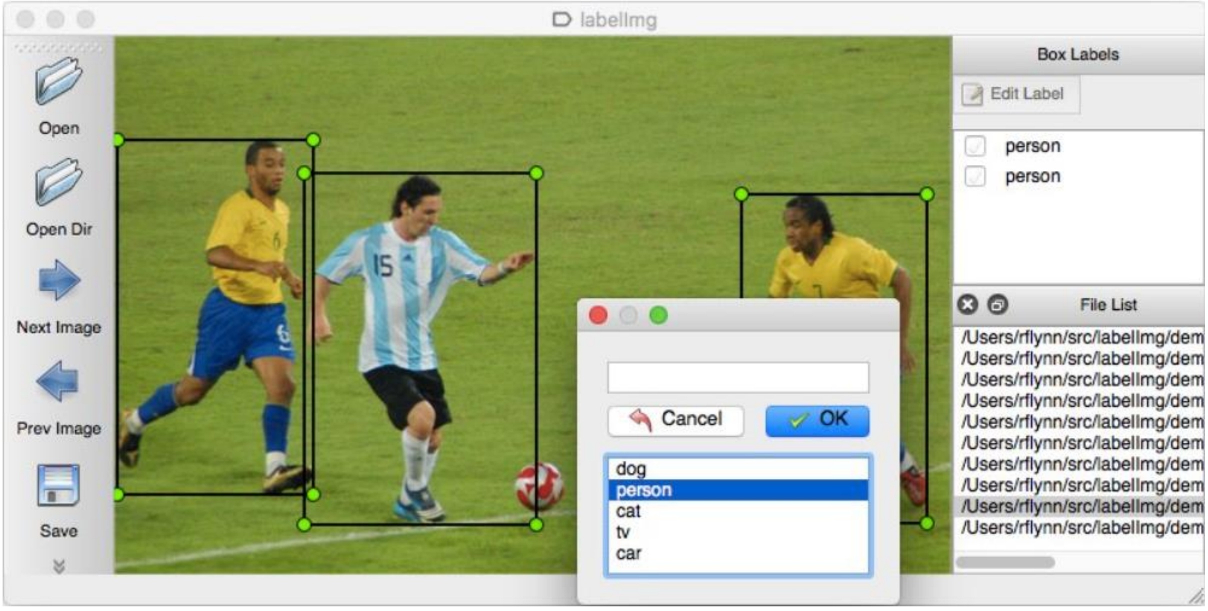

Figure 3: LabelImg Tool for Annotation

4 Google Cloud Requirements

There is a requirement to use Google Colaboratory to run the extensive model for training the YOLOv5 and YOLOv10 models. Google Colab is a free, cloud-based platform that allows to write and execute the python code in a Jupyter notebook environment. It provides access to more powerful computer resources, including GPU's, make it ideal for machine and deep learning without requiring local setup or installations. The below configuration machine can be used for performing the training of YOLOv10 and YOLOv5.

Table 3: Google Cloud Configurations

GPU Model	Specifications	Use case	Purpose
T4 GPU	16GB GDDR6, 2560 CUDA Cores	Initial training and Hyperparameter tuning	Balances speed and cost
L4 GPU	24GB GDDR6, 7680 CUDA Cores	Full data Training and Examine results	Enables faster processing than T4

5 Dataset Overview

The raw data needs to be downloaded from Mendeley data, where it is published and contributed by (Tyagi & Bansal, 2022). The twenty ISL words in the dataset are represented by RGB images of hand gestures. These words are frequently used to convey messages or request assistance in medical situations. The words in this dataset are all static sign images. Eight people between the ages of nine and thirty were photographed, six of whom were male and two of whom were female. There are 18,000 JPEG images in the dataset (Tyagi & Bansal, 2022). The URL of the data is https://data.mendeley.com/datasets/s6kgb6r3ss/2.

6 Dataset Preparation

6.1 Data Pre-processing

After extracting the data, load and execute the ‘data-prep.ipynb’ notebook in Jupyter Notebook. This code helps create the necessary directory structure for YOLO training, as illustrated in Figure 4. After this step, manually perform the label annotation task as described in Section 6.2. The final code block processes the images into an 80:20 ratio, with 14,034 images for training and 3,576 for validation. Labelling is required for both training and validation datasets.

```
├── images/          # Images folder
│   ├── train
│   └── Val
├── labels/          # Labels folder
│   ├── train
│   └── Val
```

Figure 4: Folder Structure of Data for YOLOv10 training

```
# Process the training and validation images
process_files(train_image_paths, image_train_dir, label_train_dir)
process_files(val_image_paths, image_val_dir, label_val_dir)

print("Images and labels have been processed and moved successfully.")
```

```
Images and labels have been processed and moved successfully.
```

Figure 5: Data pre-processing

6.2 Data Annotation

This is an important step in the data processing workflow, where each image is labelled by creating bounding boxes and applying the appropriate labels. The label information for each image is saved as a .txt file in the same folder. These files must be moved into the ‘labels’ folder, as referenced in Figure 4. Any errors in this process may require repeating the task, which could result in suboptimal model training outcomes. It is essential to save the annotations in YOLO format.

Example of YOLO labels.txt file:
class x_center y_center width height
0 0.5025 0.5 0.995 0.99

Create a ‘data.yaml’ file, the ‘data.yaml’ is a configuration file that specifies important information for training the model. It includes path to the training and validation datasets, the number of classes, and the name of the classes. The file ensures that the YOLO model correctly interprets the dataset structure and knows how to map predicting bounding boxes to specific object categories during training and inference. This file can be placed inside the dataset folder for easier access during model training. The dataset format required is same for training both YOLOv10 and YOLOv5.

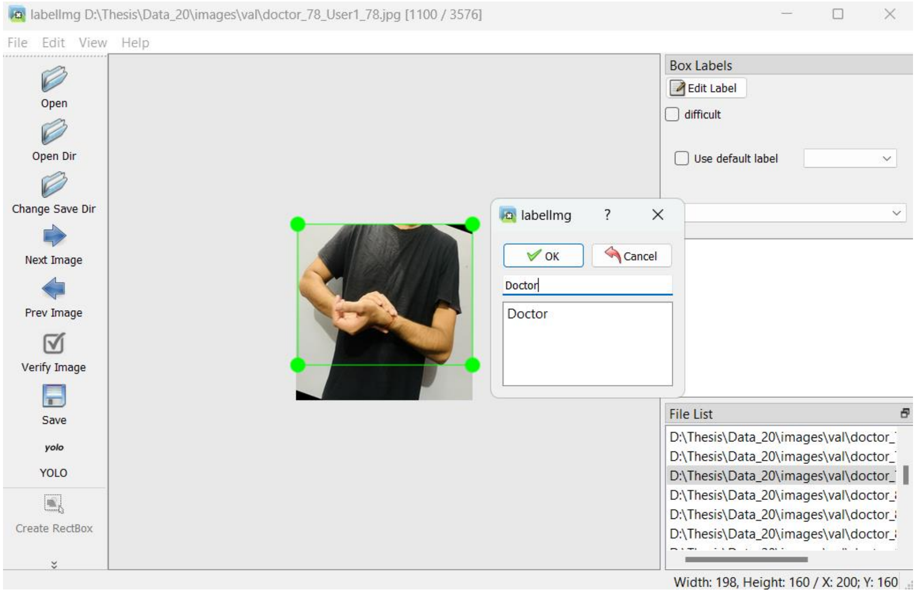

Figure 5: LabelImg Tool

```yaml
path: /content/ISL-Detection/Data # dataset root dir
train: images/train # train images (relative to 'path')
val: images/val # val images (relative to 'path')
test: images/test # test images (optional)

# Classes
names:
  0: Afraid
  1: Agree
  2: Assistance
  3: Bad
  4: Become
  5: College
  6: Doctor
  7: From
  8: Pain
  9: Pray
  10: Secondary
```

Figure 6: Format of data.yaml file

7 Model Training

YOLOv5 and YOLOv10 were chosen for their state-of-the-art performance in object detection. YOLOv5 is well-regarded for its strong community support and widespread usage, making it an excellent baseline model for comparison. YOLOv10, the latest successor, was selected for this research due to its improved accuracy and precision over previous models. The model training was conducted in two phases: first by running the YOLOv5 model, and then by running YOLOv10. The pre-processed dataset, along with the 'data.yaml' file, is a prerequisite for training and is saved in the 'data' folder on GitHub at https://github.com/ajivishwam/ISL-Detection. This dataset can be cloned and used for training.

7.1 Training Preparation Steps for YOLOv5 and YOLOv10 model

Import the notebook titled ‘ISL_Detection_yolov5.ipynb’ and ‘ISL_Detection_yolov10.ipynb’ from the ‘Notebooks’ folder on GitHub (https://github.com/ajivishwam/ISL-Detection) for training the YOLOv5 and YOLOv10 model in Google Colab. Follow these steps:

1) Install the dependencies and verify the GPU configuration within Google cloud as Shown in Figure 7 and Figure 8.

```
[ ] !nvidia-smi

Thu Aug  8 12:16:14 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   39C    P8               9W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

Figure 7: Google Colab T4 GPU

```
[ ] !pip install --upgrade ultralytics==8.2.67
    !pip install -q supervision

Collecting ultralytics==8.2.67
  Downloading ultralytics-8.2.67-py3-none-any.whl.metadata (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.3/41.3 kB 1.6 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0.0,>=1.23.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (fr
```

Figure 8: Dependency Installation

2) Clone the Github repository containing the processed data, which can be found in the 'Data' folder.

```
[ ] !git clone https://github.com/ajivishwam/ISL-Detection.git

Cloning into 'ISL-Detection'...
remote: Enumerating objects: 17338, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 17338 (delta 0), reused 4 (delta 0), pack-reused 17333
Receiving objects: 100% (17338/17338), 492.96 MiB | 39.55 MiB/s, done.
Resolving deltas: 100% (8/8), done.
Updating files: 100% (71665/71665), done.
```

Figure 9: Download processed dataset

3) Cloning the YOLOv5 (https://github.com/ultralytics/yolov5.git)and YOLOv10 (https://github.com/THU-MIG/yolov10.git) files from official repository

```
!git clone https://github.com/ultralytics/yolov5.git
%cd yolov5

Cloning into 'yolov5'...
remote: Enumerating objects: 16836, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 16836 (delta 1), reused 6 (delta 0), pack-reused 16825
Receiving objects: 100% (16836/16836), 15.57 MiB | 17.12 MiB/s, done.
Resolving deltas: 100% (11541/11541), done.
/content/yolov5
```

Figure 10: Download Yolov5 repository

4) Install all the required dependencies in yolov5 folder.

```
[ ] !pip install -r requirements.txt

Collecting gitpython>=3.1.30 (from -r requirements.txt (line 5))
  Downloading GitPython-3.1.43-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: matplotlib>=3.3 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: numpy>=1.23.5 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: opencv-python>=4.1.1 in /usr/local/lib/python3.10/dist-pack
Collecting pillow>=10.3.0 (from -r requirements.txt (line 9))
  Downloading pillow-10.4.0-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (9.2 kB)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from -r
```

Figure 11: Install dependencies

5) Download the pre-trained weights from their official repository, the choice of weights can be made depending on computational power and specific parameters required for training.

```python
import os
import urllib.request
#Create a directory for the weights in the current working directory
weights_dir = os.path.join(os.getcwd(), 'weights')
os.makedirs(weights_dir, exist_ok = True)
#urls of the weights file
urls = ["https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5l.pt",
        "https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5m.pt",
        "https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt",
        "https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5x.pt"]

#Download each file
for url in urls:
  filename = os.path.basename(url)
  filepath = os.path.join(weights_dir, filename)
  urllib.request.urlretrieve(url, filepath)
  print(f"Downloaded: {filepath}")
```

```
Downloaded: /content/yolov5/weights/yolov5l.pt
Downloaded: /content/yolov5/weights/yolov5m.pt
Downloaded: /content/yolov5/weights/yolov5s.pt
Downloaded: /content/yolov5/weights/yolov5x.pt
```

Figure 12: Download weights

7.2 Model Training YOLOv5

- The model was trained using custom data and hyperparameters for 25 epochs with a batch size of 8, which yielded best results.
- These results are saved for further testing and validation. The output from the YOLOv5 model and the weights of the best trained model are saved under folder 'runs/train/exp/weights/best.pt'.
- Save this weight for running inference and testing on web application.

```
!python train.py --imgsz 256 --batch-size 8 --epochs 25 --data '/content/ISL-Detection/Data_20/data.yaml' --weights 'weights/yolov5m.pt' \
--optimizer AdamW --patience 0 --cos-lr --nosave --cache
```

```
      22/24      1.07G   0.002803   0.003132     0.0398         20        256: 100% 1788/1788 [03:23<00:00,  8.78it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 224/224 [00:30<00:00,  7.38it/s]
                   all       3576       3576      0.913      0.928       0.97      0.867

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      23/24      1.07G   0.002613   0.002975    0.03895         25        256: 100% 1788/1788 [03:25<00:00,  8.70it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 224/224 [00:30<00:00,  7.30it/s]
                   all       3576       3576      0.925      0.939      0.978      0.874

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      24/24      1.07G   0.002473   0.002939    0.03823         29        256: 100% 1788/1788 [03:27<00:00,  8.61it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 224/224 [00:29<00:00,  7.53it/s]
                   all       3576       3576      0.934      0.951      0.981      0.881

25 epochs completed in 1.682 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 42.2MB
Optimizer stripped from runs/train/exp/weights/best.pt, 42.2MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 212 layers, 20929713 parameters, 0 gradients, 48.1 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 224/224 [00:34<00:00,  6.45it/s]
                   all       3576       3576      0.934      0.951      0.981      0.881
                Afraid       3576        180      0.934      0.983      0.988      0.897
                 Agree       3576        180      0.991          1      0.995      0.903
            Assistance       3576        180      0.919      0.989      0.994      0.909
                   Bad       3576        180      0.983      0.954      0.991      0.918
                Become       3576        180      0.966      0.939      0.988      0.887
               College       3576        180        0.9      0.961      0.979      0.877
                Doctor       3576        150      0.961      0.991      0.994      0.888
                  From       3576        180      0.979      0.983      0.995      0.898
                  Pain       3576        180          1      0.986      0.994      0.895
                  Pray       3576        180      0.973      0.994      0.995      0.895
             Secondary       3576        180      0.921      0.839      0.951      0.835
                  Skin       3576        180      0.967      0.991      0.994      0.895
                 Small       3576        180          1      0.997      0.995      0.894
              Specific       3576        180      0.965      0.762      0.954      0.827
                 Stand       3576        180      0.866      0.939      0.978      0.871
                 Today       3576        180      0.973      0.983      0.995      0.894
```

Figure 13: Model Training results

7.3 Model Training YOLOv10

- The model was trained using custom data and hyperparameters for 25 epochs with a batch size of 8, which yielded best results. These results are saved for further testing and validation.
- The output from the YOLOv10 model and the weights of the best trained model are saved under folder 'runs/detect/train/weights/best.pt'.
- Save this weight for running inference and testing on web application.

```
!yolo task=detect mode=train epochs=25 batch=8 plots=True imgsz=256 model='weights/yolov10m.pt' data='/content/ISL-Detection/Data_20/data.yaml' \
lr0=0.001 optimizer=AdamW \
hsv_h=0.015 hsv_s=0.7 hsv_v=0.4 \
degrees=5.0 translate=0.1 scale=0.5 shear=2.0 mosaic=1.0 mixup=0.2 \
patience=10 warmup_epochs=5.0 cos_lr=True amp=True \
save_period=1
```

```
New https://pypi.org/project/ultralytics/8.2.70 available 😃 Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.2.67 🚀 Python-3.10.12 torch-2.0.1+cu117 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=weights/yolov10m.pt, data=/content/drive/MyDrive/Colab Note
Overriding model.yaml nc=80 with nc=20

                   from  n    params  module                                       arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv             [3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      23/25      1.16G    0.05964    0.09574      1.818          5        256: 100% 1789/1789 [05:14<00:00,  5.69it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 224/224 [00:37<00:00,  5.94it/s]
                   all       3576       3576      0.985      0.999      0.994      0.994

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      24/25      1.13G    0.05779    0.08852      1.817          5        256: 100% 1789/1789 [05:15<00:00,  5.68it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 224/224 [00:38<00:00,  5.87it/s]
                   all       3576       3576       0.99      0.996      0.994      0.994

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      25/25      1.16G    0.05489    0.08517      1.819          5        256: 100% 1789/1789 [05:14<00:00,  5.69it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 224/224 [00:39<00:00,  5.72it/s]
                   all       3576       3576      0.995      0.992      0.994      0.994

25 epochs completed in 2.643 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 33.5MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 33.5MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.2.67 🚀 Python-3.10.12 torch-2.0.1+cu117 CUDA:0 (Tesla T4, 15102MiB)
YOLOv10m summary (fused): 369 layers, 16,473,544 parameters, 0 gradients, 63.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 224/224 [00:37<00:00,  5.90it/s]
                   all       3576       3576       0.99      0.996      0.994      0.994
                Afraid        180        180      0.997          1      0.995      0.995
                 Agree        180        180          1          1      0.995      0.995
            Assistance        180        180          1          1      0.995      0.995
                   Bad        180        180          1          1      0.995      0.995
                Become        180        180      0.997          1      0.995      0.995
               College        180        180      0.997          1      0.995      0.995
                Doctor        150        150          1      0.992      0.995      0.995
                  From        180        180      0.997          1      0.995      0.995
                  Pain        180        180      0.997          1      0.995      0.995
                  Pray        180        180          1          1      0.995      0.995
             Secondary        180        180      0.997          1      0.995      0.995
                  Skin        180        180          1          1      0.995      0.995
                 Small        180        180      0.997          1      0.995      0.995
              Specific        180        180          1          1      0.995      0.995
                 Stand        180        180      0.993          1      0.995      0.995
                 Today        180        180      0.999          1      0.995      0.995
                  Warn        180        180      0.957       0.95      0.987      0.987
                 Which        180        180          1          1      0.995      0.995
                  Work        186        186      0.997          1      0.995      0.995
                   You        180        180      0.884      0.978       0.99       0.99
Speed: 0.1ms preprocess, 3.2ms inference, 0.0ms loss, 0.2ms postprocess per image
Results saved to runs/detect/train2
```

Figure 14: Model Training results of YOLOv10

8 Running Inference

The next step is to run the inference by testing the model on new, previously untested images. These test images should be self-created and should include images that were not used during training.

8.1 Validation Testing on YOLOv5

To validate the model, follow these steps:

- Run the inference code for detection as shown in the referenced Figure 10.
- Provide path to the ‘best.pt’ weight saved during model training, as well as the path to the new images.
- The inference results will be saved in the ‘runs/detect/exp’ directory.

```
## Inference or detection on new images
!python detect.py --source '/content/ISL-Detection/Data_20/images/test' \
--weights '/content/yolov5/runs/train/exp/weights/best.pt' --img 640 --save-txt --save-conf
```

```
Fusing layers...
Model summary: 212 layers, 20929713 parameters, 0 gradients, 48.1 GFLOPs
image 1/45 /content/test_images/Image 1.jpeg: 640x640 1 From, 901.8ms
image 2/45 /content/test_images/Image 10.jpeg: 640x640 1 Bad, 1 Become, 1 College, 845.6ms
image 3/45 /content/test_images/Image 11.jpeg: 640x640 (no detections), 816.0ms
image 4/45 /content/test_images/Image 12.jpeg: 640x640 (no detections), 842.0ms
image 5/45 /content/test_images/Image 13.jpeg: 640x640 1 Become, 1 Work, 840.9ms
image 6/45 /content/test_images/Image 14.jpeg: 640x640 1 Pain, 881.5ms
image 7/45 /content/test_images/Image 15.jpg: 640x640 1 Become, 837.4ms
image 8/45 /content/test_images/Image 16.jpg: 640x640 1 Afraid, 1 Become, 1 You, 855.6ms
image 9/45 /content/test_images/Image 17.jpg: 640x640 2 Afraids, 1 Bad, 1 You, 833.1ms
image 10/45 /content/test_images/Image 18.jpg: 640x640 1 Agree, 2 Froms, 860.7ms
image 11/45 /content/test_images/Image 19.jpg: 640x640 1 Which, 1 You, 840.2ms
image 12/45 /content/test_images/Image 2.jpeg: 640x640 1 Pain, 1235.4ms
image 13/45 /content/test_images/Image 20.jpg: 640x640 2 Becomes, 1395.4ms
image 14/45 /content/test_images/Image 21.jpg: 640x640 2 Becomes, 1385.9ms
image 15/45 /content/test_images/Image 22.jpg: 640x640 1 From, 1 Pain, 1 Work, 1250.1ms
image 16/45 /content/test_images/Image 23.jpg: 640x640 1 Bad, 1 Become, 836.5ms
image 17/45 /content/test_images/Image 24.jpg: 640x640 2 Becomes, 1 College, 1 Pain, 833.9ms
image 18/45 /content/test_images/Image 25.jpg: 640x640 1 Become, 1 Pain, 830.6ms
image 19/45 /content/test_images/Image 26.jpg: 640x640 1 Become, 1 You, 831.6ms
image 20/45 /content/test_images/Image 27.jpg: 640x640 1 Doctor, 836.7ms
```

Figure 15: Inference results of Yolov5

8.2 Validation Testing on YOLOv10

To validate the model, follow these steps:

- Run the inference code for detection as shown in the referenced Figure 10.
- Provide path to the ‘best.pt’ weight saved during model training, as well as the path to the new images.
- The inference results will be saved in the ‘runs/detect/predict’ directory.

```
!yolo task=detect mode=predict conf=0.70 save=True model='/content/yolov10/runs/detect/train/weights/best.pt' source='/content/ISL-Detection/Data_20/images/test'
```

```
Ultralytics YOLOv8.2.75 🚀 Python-3.10.12 torch-2.0.1+cu117 CUDA:0 (Tesla T4, 15102MiB)
YOLOv10m summary (fused): 369 layers, 16,473,544 parameters, 0 gradients, 63.5 GFLOPs

image 1/45 /content/ISL-Detection/Data_20/images/test/Image 1.jpeg: 256x256 (no detections), 23.2ms
image 2/45 /content/ISL-Detection/Data_20/images/test/Image 10.jpeg: 256x256 1 Pain, 21.6ms
image 3/45 /content/ISL-Detection/Data_20/images/test/Image 11.jpeg: 256x256 1 Pain, 18.6ms
image 4/45 /content/ISL-Detection/Data_20/images/test/Image 12.jpeg: 256x256 1 From, 18.9ms
image 5/45 /content/ISL-Detection/Data_20/images/test/Image 13.jpeg: 256x256 1 From, 18.8ms
image 6/45 /content/ISL-Detection/Data_20/images/test/Image 14.jpeg: 256x256 1 From, 18.8ms
image 7/45 /content/ISL-Detection/Data_20/images/test/Image 15.jpg: 256x256 1 Warn, 18.7ms
image 8/45 /content/ISL-Detection/Data_20/images/test/Image 16.jpg: 256x256 1 Afraid, 18.2ms
image 9/45 /content/ISL-Detection/Data_20/images/test/Image 17.jpg: 256x256 1 Afraid, 18.5ms
image 10/45 /content/ISL-Detection/Data_20/images/test/Image 18.jpg: 256x256 1 Agree, 18.0ms
image 11/45 /content/ISL-Detection/Data_20/images/test/Image 19.jpg: 256x256 1 Agree, 19.4ms
image 12/45 /content/ISL-Detection/Data_20/images/test/Image 2.jpeg: 256x256 1 Today, 18.9ms
image 13/45 /content/ISL-Detection/Data_20/images/test/Image 20.jpg: 256x256 1 Assistance, 19.3ms
image 14/45 /content/ISL-Detection/Data_20/images/test/Image 21.jpg: 256x256 1 Assistance, 20.9ms
image 15/45 /content/ISL-Detection/Data_20/images/test/Image 22.jpg: 256x256 1 Bad, 20.5ms
```

Figure 16: Inference results of Yolov10

```python
import os
from IPython.display import Image, display

# Specify the directory containing images
directory_path = '/content/yolov10/runs/detect/predict/'

# List all image files in the directory
image_files = [f for f in os.listdir(directory_path) if f.endswith(('png', 'jpg', 'jpeg'))]

# Display each image
for image_file in image_files:
    image_path = os.path.join(directory_path, image_file)
    display(Image(filename=image_path))
```

Figure 16: Display Inference Image

9 Indian Sign Language Recognition and Translation text to speech Flask web application

This project is an Indian Sign Language Detection web application that allows users to upload static sign images, or use their webcam to perform Indian Sign Language (ISL) detection using YOLO models (YOLOv5 or YOLOv10). The detected signs can be translated into various Indian languages, and the translated text can be converted to speech. Some of the features are listed below:

- Upload static sign images or use the webcam for sign detection
- Choose between YOLOv5 and YOLOv10 models
- Translate detected text into various Indian languages
- Text-to-speech conversion for translated text
- Go back button to clear results

Steps to replicate the installation to test the application:

1. Clone the GitHub repository (https://github.com/ajivishwam/Flash-ISL-WebApp.git)

```
git clone https://github.com/ajivishwam/Flash-ISL-WebApp.git
cd Flash-ISL-WebApp
```

2. Install the required dependencies

```
pip install -r requirements.txt
```

3. Create a virtual environment for FinalYOLOv5. Run the below script using Git Bash or shell that support `.sh` files

```
cd FinalYOLOv5/
.setup.sh
```

4. Create a virtual environment for FinalYOLOv10. Run the below script using Git Bash or shell that support `.sh` files

```
cd FinalYOLOv10/
.setup.sh
```

5. The implementation code is in ‘app.py’. Run the Python application to serve it locally at ‘http://127.0.0.1:5000’.
6. Upload images by selecting the model and translation language, then upload the static sign images, as shown in Figure 17.
7. The ‘result.html’ is generated, displaying the output of the images, as illustrated in Figures 18 and 19.
8. Figure 20 shows the predictions of ISL Image detection and their translation to text and speech for multiple images.

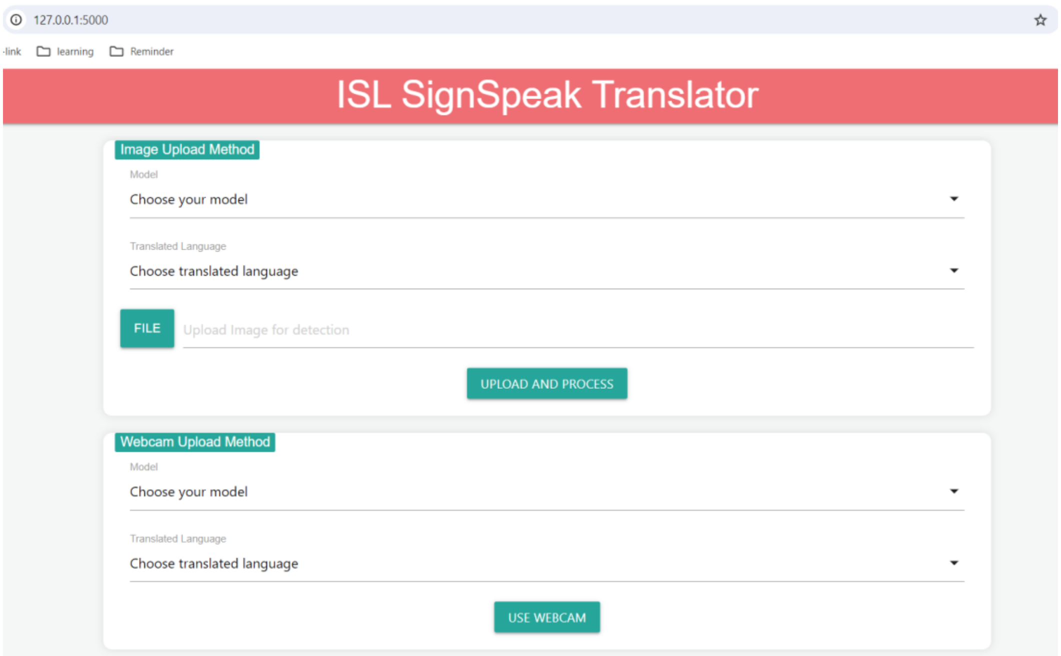

Figure 16: Home page of ISL Application

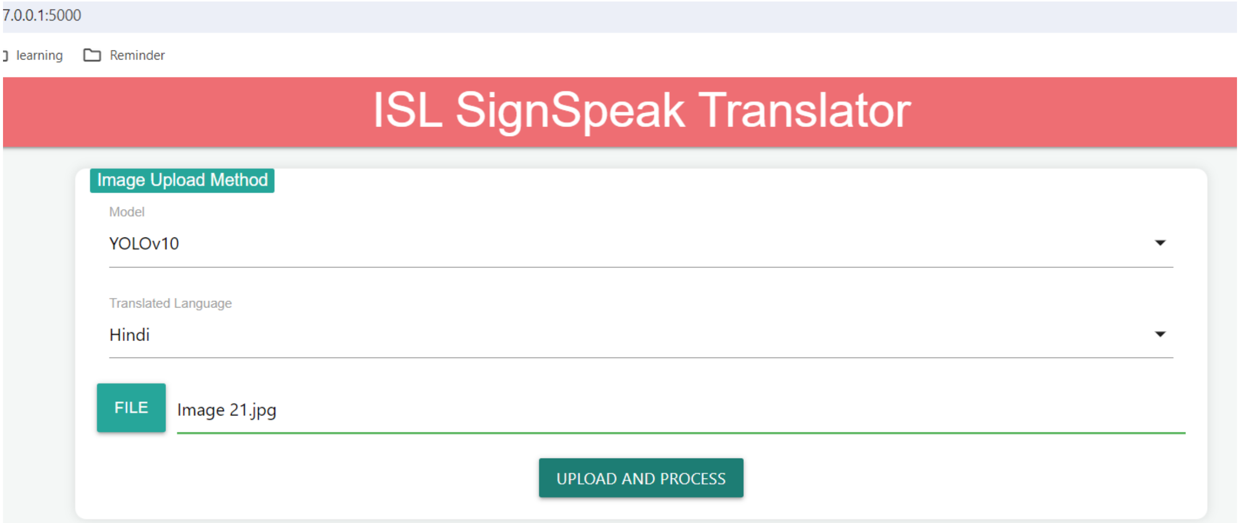

Figure 17: ISL Image upload and Model Selection

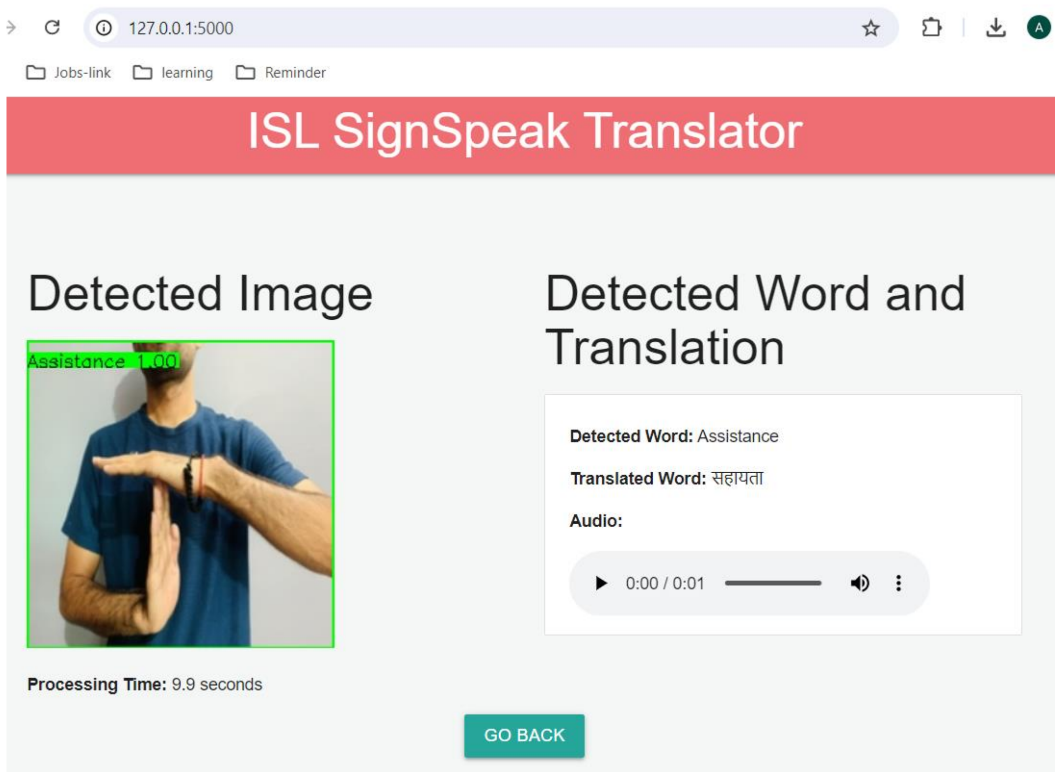

Figure 18: ISL Image Prediction, translation with audio

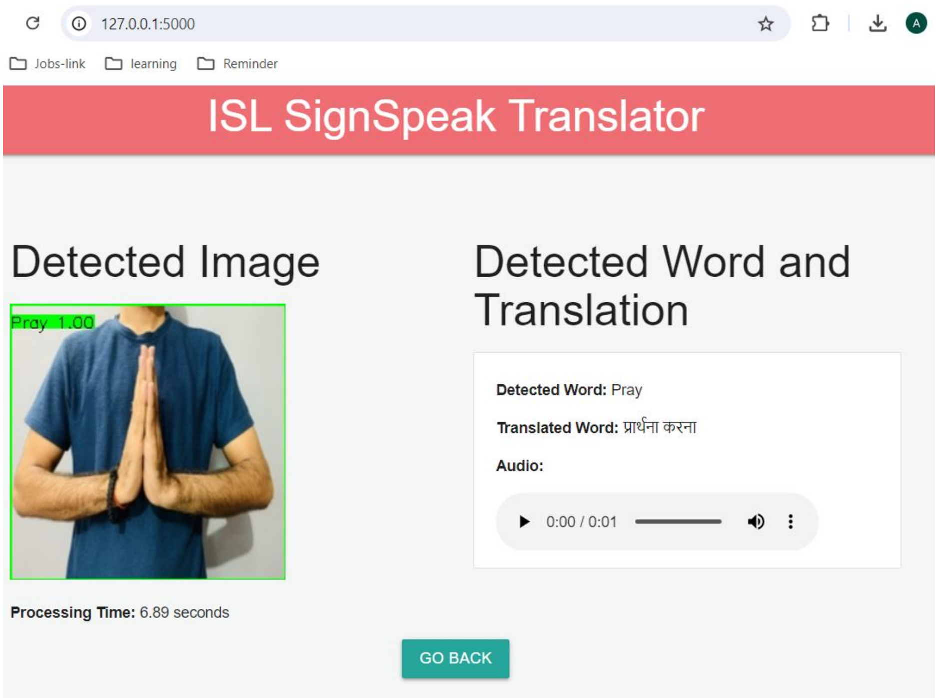

Figure 19: ISL Image Prediction and translation with audio

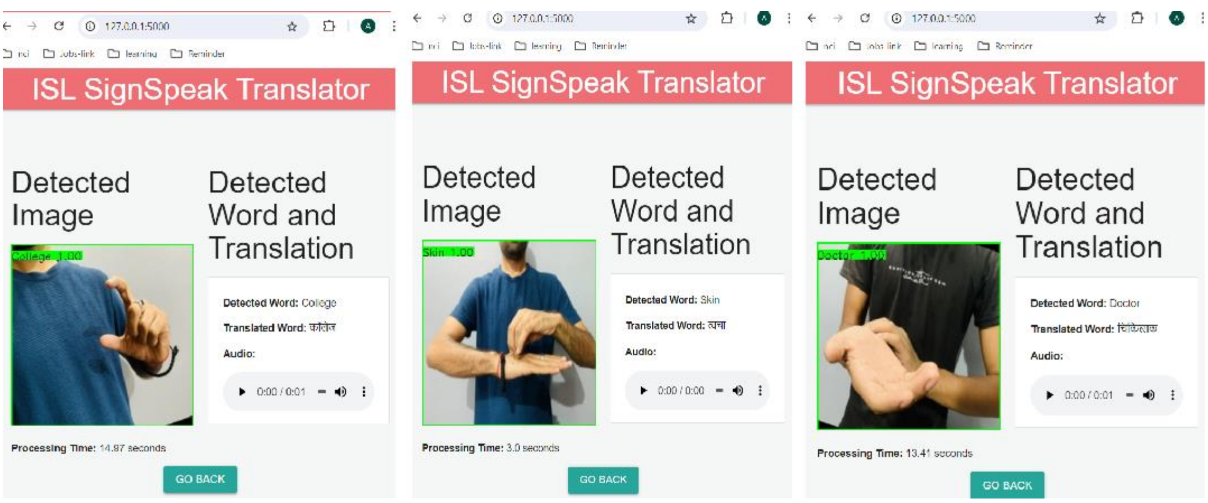

Figure 20: ISL Image Predictions

References

Tyagi, A., & Bansal, S. (2022). *Indian sign Language-Real-life Words*. Retrieved from https://data.mendeley.com/datasets/s6kgb6r3ss/2