American Journal of Artificial Intelligence
2023; 7(2): 40-51
http://www.sciencepublishinggroup.com/j/ajai
doi: 10.11648/j.ajai.20230702.12
ISSN: 2639-9717 (Print); ISSN: 2639-9733 (Online)

Signed Language Translation into Afaan Oromo Text Using Deep-Learning Approach

Diriba Negash Tesso*, Etana Fikadu Dinsa, Hawi Fikadu Kenani

Department of Computer Science, College of Engineering and Technology, Wallaga University, Nekemte, Ethiopia

Email address:

diribanegash84@gmail.com (Diriba Negash Tesso), fikaduetana@yahoo.com (Etana Fikadu Dinsa), hawif2014@gmail.com (Hawi Fikadu Kenani)

*Corresponding author

To cite this article:

Diriba Negash Tesso, Etana Fikadu Dinsa, Hawi Fikadu Kenani. Signed Language Translation into Afaan Oromo Text Using Deep-Learning Approach. *American Journal of Artificial Intelligence*. Vol. 7, No. 2, 2023, pp. 40-51. doi: 10.11648/j.ajai.20230702.12

Received: October 17, 2023; **Accepted**: November 2, 2023; **Published**: November 17, 2023

Abstract: A person who is unable to talk or hear anything can communicate via sign language. For those who have trouble hearing, sign language is a great way to communicate their thoughts and feelings. The vocabulary, grammar, and allied lexicons of sign language are well- defined. This study focuses primarily on Signed Afaan Oromo. The main issue in our society is the detection of Sign Language for the Afaan Oromo language. The construction of static word level, alphabet, and number translations into their equivalent Afaan Oromo text is the main focus of this thesis study. Video frames are used as the system's input, and Afaan Oromo text is used as the system's ultimate output. Data from 90 classes at the alphabet, number, and word level from five special needs instructors have been collected as part of an experiment and literature study to help answer the research objectives. Preprocessing, such as frame extraction, resizing, labeling, and splitting data using Roboflow, as well as the conversion of photos into Yolo model format, was done in order to train our model. Finally, based on the results of our experiment, we can quickly and effectively recognize and classify gestures using data sets of a medium size. The image, webcam, and video file's promising value and forecast results indicate that the yolov5 algorithm has a good chance of successfully detecting the sign in real-time. We trained and tested the model using a signed Afaan Oromo dataset. The YOLOv5s model was successful in obtaining accuracy of 90%, recall of 92.5%, mAP of 93.2% at 0.5 IoU, and a score of 71.5% at 0.5:0.95 IoU, which is suitable for real-time gesture translation.

Keywords: Signed Language, Deep Learning, Computer Vision, CNN, YOLOv5

1. Introduction

A certain nation or region employs a communication system that includes a predetermined set of written symbols and echoes for speaking or script [1, 20]. Currently, Afaan Oromo is the official language of the regional state of Oromia and is used in offices, as a medium of instruction for all non-language courses from elementary school through higher education (university), and as the language of the federal government. The fundamental building blocks of a language in the Afaan Oromo writing system are the alphabet, numbers, and words. Phrases, clauses, and sentences are created through the combination of these linguistic building blocks [3]. Languages that convey meaning using a visual-manual modality are known as sign (signed) languages [4, 5]. A sign language is a language where gestures and facial expressions are used to transmit information instead of vocal tracts, making it more careful than a spoken language where vocal tracts are employed. Genetic factors, problems after delivery, or other infectious disorders can also contribute to hearing impairment [6]. People with hearing loss have simple needs, just like other people, such as learning, teaching, reading, writing, and communicating, albeit these activities may not be completely stress-free for them. The information that enables a computer to detect the sign used by a signer and translate it into text using an algorithm is known as sign language translation [7-9]. By helping hearing-impaired people communicate with other communities and overcome

their communication barriers, artificial intelligence technologies can make a significant contribution to their social inclusion [8, 10]. We are focused on signed language translation into Afaan Oromo text that is expressed in the form of static sign language of word level that indicates (one or two hands that represents a word) and finger vocabulary (One gesture represents each alphabet (a-z) and numbers (0-10). we disregarded sentence level in Afaan Oromo sign language and Translation from signed language to text merely; it does not incorporate translation from text to sign.

2. Methodology

We go into great detail about the study's architectures, dataset preparation, preprocessing, training models, detection, and classification procedures. The following techniques are utilized to accomplish the study's goals [11, 12]. The class designates the various AO Language alphabets, numbers, and words that the network has been trained on [2]. The unique Yolov5 structures that are integrated to assess and recognize signs are part of the suggested model Figure 1 [11]. The sign language fingerspelling and upper body dataset is used.

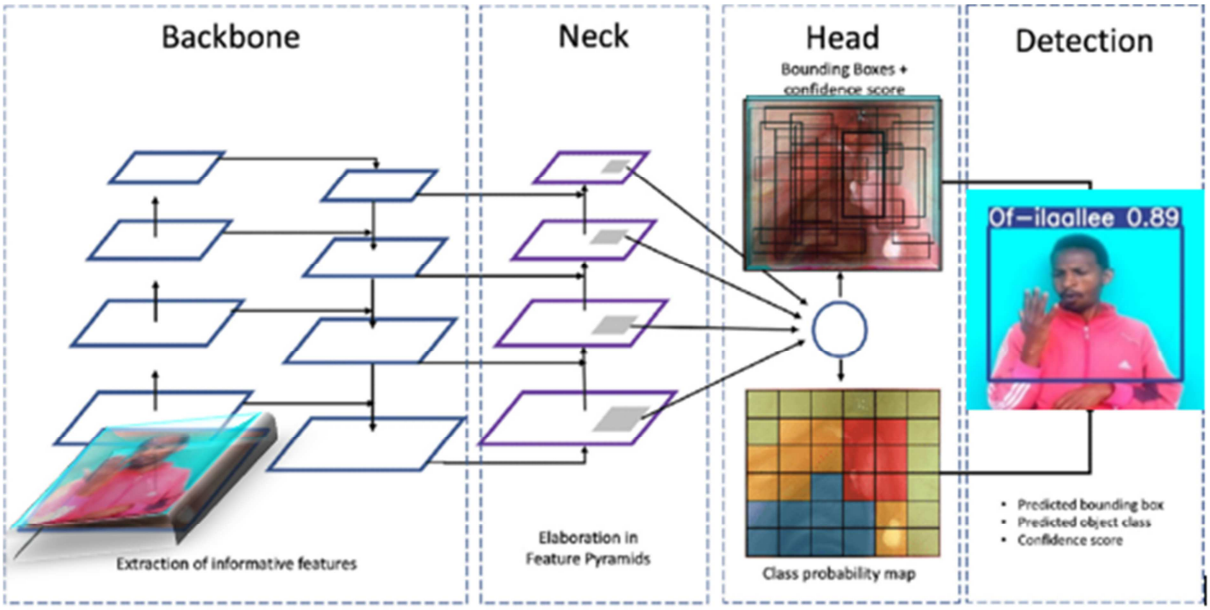

Figure 1. *General diagram of proposed system.*

The following would be the typical implementation steps to train the model using the signed language dataset.

2.1. Dataset

The datasets used for model experimentation, the programming language used to create the model, and the evaluation techniques used to gauge the effectiveness of our sign detector and recognizer are all covered in the following section. Data that is required to perform the research has been gathered for 90 classes in order to achieve this. However, using a mobile camera, from fives instructor recorded 90 Latin-script signed languages, specifically, for words 54, numerals (0–10), and alphabets (A–Z).

2.2. YOLOv5 Algorithm

YOLO is typically faster than other object detection algorithms [12, 14]. To do this, YOLO divides an image into a grid, and then classifies and localizes each piece of the grid. It then forecasts the location of bounding boxes. Instead of classification-based methods, regression-based algorithms are used to predict these bounding boxes [13]. Typically, classification-based algorithms involve two steps: first, choosing the region of interest, and second, using CNN to detect items in the regions chosen [15]. In this work, we use the most recent Yolov5 implementation created by Ultralytics to perform on a signed language dataset [12]. We trained our own model using transfer-learning techniques, assessed its effectiveness, used it to inference, and even converted it to different file formats like ONNX and TensorRT [14]. The following model architecture applies to YOLOv5 [16, 17].

Three building elements make up our model: the CSPDarknet backbone, the PANet neck, and the Yolo Layer head [13]. Before being sent to PANet for feature fusion, the data are first fed to CSPDarknet for feature extraction. Yolo Layer then outputs the results of the detection (class, score, position, and size) [13, 15].

Model Backbone: - is frequently used to extract crucial details from an input image. The Cross Stage Partial networks are the foundation of YOLOv5's feature extraction process for highly informative features from an input image [12, 13]. With deeper networks, CSPNet has demonstrated a considerable reduction in processing time Figure 2.

Model Neck: - The fundamental function of the model neck is to produce feature pyramids. Models that use feature pyramids scale objects well in general [18]. The ability to recognize the same thing in various sizes and scales is helpful. Models that use feature pyramids perform well on unobserved data [19, 20]. We obtained feature pyramids using PANet as

our neck. The network neck used by Yolov5 for feature extraction is the PANet FPN or feature pyramid network [21].

Model Head: - is mostly employed to carry out the final detecting step. It applied anchor boxes to the features and produced final output vectors that included bounding boxes, class probabilities, and object-ness scores [22].

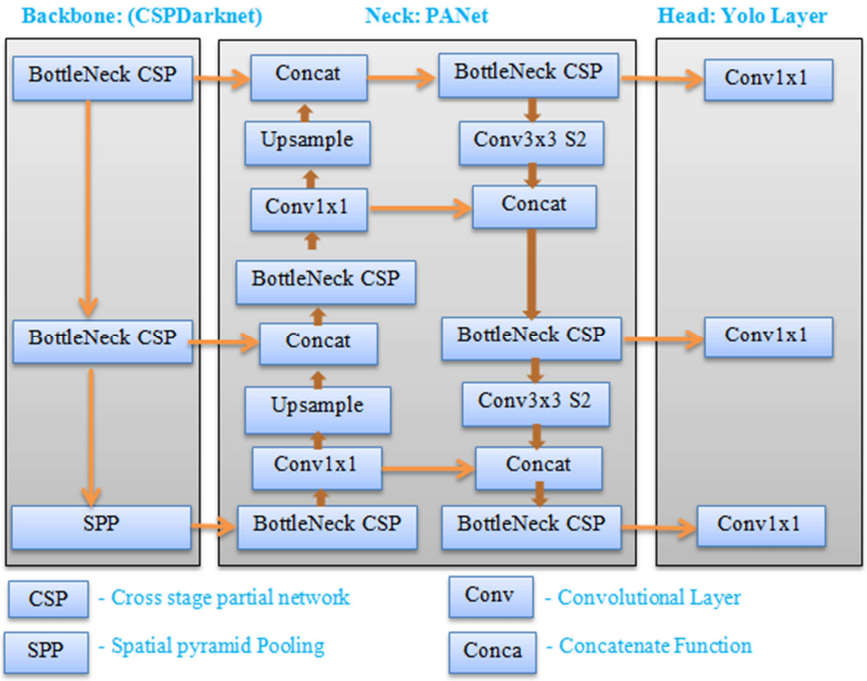

Figure 2. *YOLOv5 network architecture [33].*

3. Results and Discussion

Traditional approaches for sign language translation have been investigated for a long time [23, 41]. These techniques have demonstrated utility in their specific fields and have produced sufficient evidence of their effectiveness and accuracy [24, 5]. Deep learning's golden age has only begun, though. The objective of the current work is to use deep learning to improve the accuracy of sign language translation [25]. Deep learning techniques are only just beginning to be applied to sign language translation. This study places a strong emphasis on the use of deep learning techniques, in particular Yolov5 [18]. The use of Yolov5 as the foundation for sign language translation is the focus of the research question [26]. An original way to approach the issue is to see the translation process as a work of object detection and categorization.

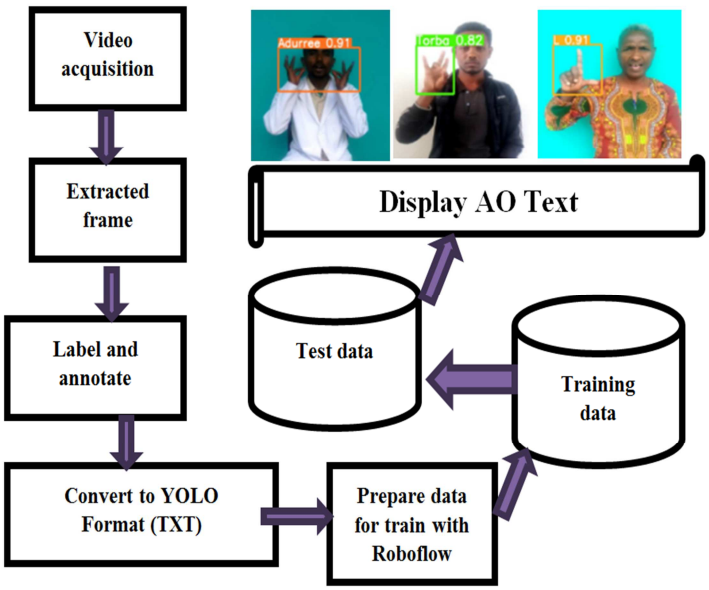

Figure 3. *Workflow of translation Trained Model.*

3.1. Environment Tools

A free Integrated Development Environment from Google, Google Colab supports AI research and education. Colaboratory offers the same coding environment as Jupyter Notebook, and Tensor Process Unit and Graphic Process (GPU) usage are also free (TPU) [15]. Deep learning research libraries including PyTorch, TensorFlow, Keras, and OpenCV are already installed on Google Colab. Since standard computers lack GPU, machine learning or deep learning methods require a system to have high speed and processing capability [15, 27]. To help AI researchers, Colab offers GPU (Tesla V80 and TPU) on cloud, one of the most powerful GPUs available right now. Colab offers a 150GB primary drive and 25GB RAM [28]. The Yolov5 architecture was developed conceptually and published via a GitHub project.

```
!git clone https://github.com/ultralytics/yolov5  # clone repo
!pip install -qr yolov5/requirements.txt  # install dependencies (ignore errors)
%cd yolov5

import torch  # Pytorch
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.')

Setup complete.
```

Figure 4. *Cloning and installing the YOLOv5 repository.*

3.2. Dataset Preparation for Training

Since Colab is a Google service, it allows linking to a personal Google drive account to get an api key from the drive for use in training the model and saving the results later. In this phase, we exported our dataset to Yolov5, which split data into three (train, validation, and test) using the Roboflow open source [11].

```
# Export code snippet and paste here
%cd /content
#!curl -L "https://app.roboflow.com/ds/hNiiNVOmR4?key=bJ95GJt6qB" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
!curl -L "https://app.roboflow.com/ds/Y2e5icQxkL?key=fLW5joKNUn" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip

Streaming output truncated to the last 5000 lines.
 extracting: train/labels/r43_jpg.rf.f451f12417bbfe19892fe72cfa111e45.txt
 extracting: train/labels/r44_jpg.rf.51ac4a40593c6a26e753976341cb8d4b.txt
 extracting: train/labels/r44_jpg.rf.7dfc410909c24aefed62ec38c48543d2.txt
 extracting: train/labels/r44_jpg.rf.b8bbd3610bf86a99a5af73e811d840a3.txt
 extracting: train/labels/r46_jpg.rf.913c63e1c616c1bcabd202931fb7fa15.txt
```

Figure 5. *Importing dataset from Roboflow.*

The above data file is a common format for bounding boxes in the object detection dataset. The bounding boxes data in YOLO is formatted as (class, X_{center}, Y_{center}, width, height). With minimum point (x_{min}, y_{min}), the center point (x_{center}, y_{center},) can be calculated as follow [29]:

$$X_{center} = X_{min} + \frac{width}{2}$$

$$X_{center} = X_{center} + \frac{height}{2}$$

3.3. Training of Model

We utilized the yolov5s model, which was the fastest, to train our data [13]. These architectures will be read from the yaml file by the YOLOv5 model on PyTorch, which will then create them in the train.py file. Additionally, this makes it simpler to modify the architecture in accordance with the various object detection issues. Additionally, we used the model's pre-trained weights for transfer learning rather than training fresh weights from scratch, which takes a lot of time and requires too much computing power to train on a laptop. 100 training epochs were utilized to train the model with a batch size of "16." Here, we have a lot of parameters that we can pass [12, 29, 30]:

Since the purpose of this thesis is to assess the performance of the Yolov5 algorithm, the original architecture will be used and the model won't be temporarily configured or enhanced with additional algorithms or optimization techniques. The number of classes must be changed for the signed AO dataset. The anchor box auto-learning has now been integrated with the anchor box settings, and it is now default to ignoring them.

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

# parameters
nc: 80  # 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, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, C3, [1024, False]],  # 9
  ]
```

Figure 6. *Sample YOLOv5 architecture provided by Ultralystic.*

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

```
In [5]: # this is the YAML file Roboflow wrote for us that we're loading into this notebook with our data
        %cat data.yaml

        train: ../train/images
        val: ../valid/images

        nc: 90
        names: ['A', 'Abbaa Manaa', 'Abbaa', 'Adurree', 'Afur', 'Akkam', 'Amma', 'B', 'Bicuu', 'Bilbila', 'Bishaan', 'Booda', 'Boor',
        'C', 'D', 'Dallansuu', 'Dansaa', 'Dheeraa', 'Dhukkubsataa', 'Dibaabee', 'Dulloomaa', 'Duwwaa', 'E', 'F', 'Fagoo', 'Funyaan', 'F
        urdaa', 'Furtuu', 'G', 'Gabaabaa', 'Gadduu', 'Gubbaa', 'Guutuu', 'H', 'Haadha Warraa', 'Haadha', 'Har-a', 'Hiriyyaa', 'I', 'Ij
        a', 'Ilkaan', 'Isa', 'Isaan', 'Ishee', 'J', 'Jaha', 'Jala', 'K', 'Kaleessa', 'Kam', 'Keessa', 'Kitaaba', 'Kudhan', 'L', 'Lama',
        'Leenca', 'Loltuu', 'M', 'Maatii', 'Mana', 'Muka', 'Muuzii', 'N', 'Nuyi', 'O', 'Obboleessa', 'Obboleettii', 'Of-ilaallee', 'P',
        'Q', 'Qalama', 'Qeerramsa', 'Qotiyyoo', 'Qulqulluu', 'Qurxummii', 'R', 'S', 'Saddeet', 'Sadii', 'Sagal', 'Shan', 'T', 'Tokko',
        'Torba', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

Figure 7. *Overwrite the number of classes and save as a model.*

The model will be trained by compiling file train.py together with its programmable arguments using the command line displayed in Figure 8.

```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 100 --data '../data.yaml' --cfg ./models/custom_yolov5s.yaml --weights 'yolov5s.pt
```

Figure 8. *Implement the training process.*

The following are Parameters lists: Batch - batch size (16), Epochs - number of epochs (100), Data - path to the data-configurations file, Weights - path to initial weights, cfg - path to the model-configurations file (*yolo5s.yaml, yolov5m.yaml, yolov5l.yaml, yolov5x.yaml*), Cache - cache images for faster training and Img - image size in pixels (416) [13].

```
Epoch   gpu_mem       box       obj       cls    labels  img_size
97/99     2.09G   0.02206  0.008894   0.01108        18       416: 100% 661/661 [01:49<00:00,  6.03it/s]
          Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 46/46 [00:06<00:00,  6.78it/s]
            all       1442       1430      0.908      0.929      0.932       0.71

Epoch   gpu_mem       box       obj       cls    labels  img_size
98/99     2.09G   0.02195  0.008902   0.01104        31       416: 100% 661/661 [01:50<00:00,  6.00it/s]
          Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 46/46 [00:06<00:00,  6.78it/s]
            all       1442       1430      0.908      0.929      0.932      0.711

Epoch   gpu_mem       box       obj       cls    labels  img_size
99/99     2.09G   0.02161  0.008823   0.01094        24       416: 100% 661/661 [01:49<00:00,  6.01it/s]
          Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 46/46 [00:06<00:00,  6.76it/s]
            all       1442       1430      0.907       0.93      0.932      0.711
```

Figure 9. *Training progress in 100 epochs.*

The average time to complete the training process on 16 batches was 10 seconds, as is shown in Figure 9, and the same goes for evaluation. The dataset had 11,993 images, and the total execution time was 3hr 17min 35s for 100 epochs. The last epoch's mAP is 93 percent. The weighting result from the previous epoch is not always the weight for the highest accuracy in this case. Performance metrics and training losses are saved to Tensorboard. After training is finished, the results file is plotted as a “png”, as seen in Figure 10 and 11 below.

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

Figure 10. *Use TensorBoard to load the entire training process saved in the runs folder.*

The trained model will be saved in your "weights" folder or directory once the training is finished, and the validation

metrics will be logged onto Tensorboard. Let's review the YOLOv5 losses and metrics in order to better comprehend the outcomes. The YOLO loss function is divided into three sections: box_loss, obj_loss and cls_loss [5, 29].

Recall measures how much of the true bbox was properly predicted (TP/TP + FN), while Precision evaluates how much of the bbox predictions are correct (TP/TP + FP). "mAP 0.5" refers to the mAP at a 0.5 IoU threshold. The average mAP over various IoU thresholds, ranging from 0.5 to 0.95 as shown below, is given as "mAP 0.5:0.95" Figure 10 [30].

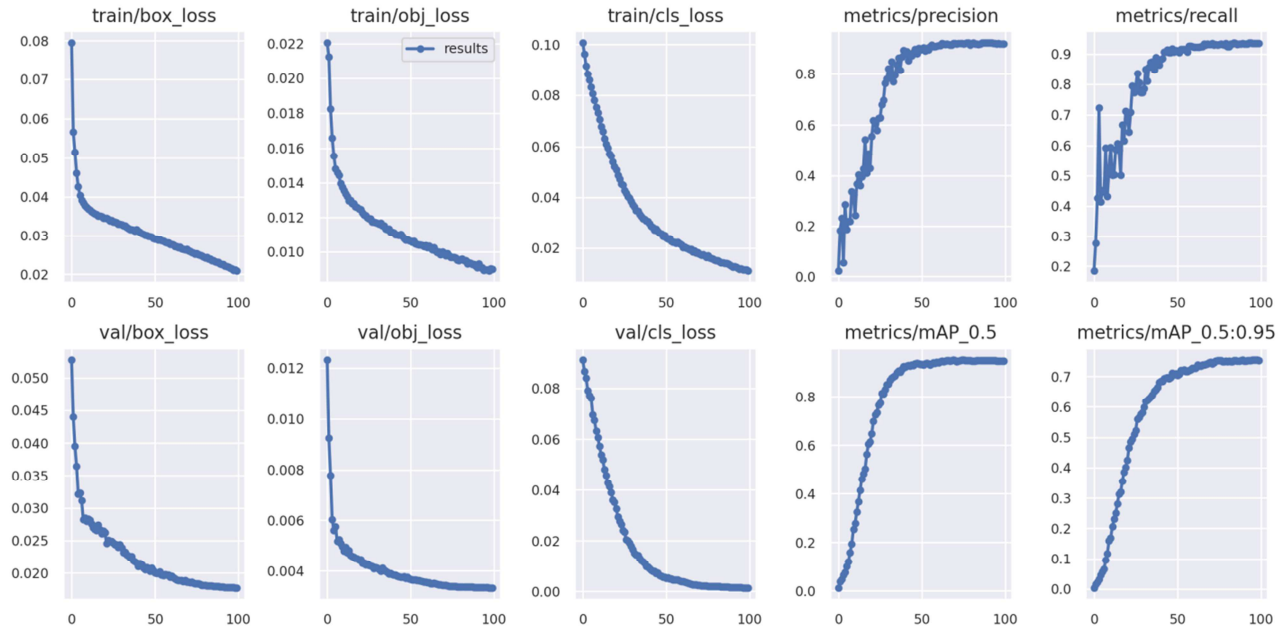

Figure 11. *The mAP, Precision, Recall and Losses, 100 epochs were visualized in graphs.*

As seen in Figure 9, the model needs around 3 hours 17 minutes and 35 seconds to finish training for 100 epochs, and the accuracy of the model is about 93%. This demonstrates that the model is quick and accurate using just the YOLOv5 original architecture, without the use of any optimization techniques. Additionally, the weighting results are saved by the model in a.pt file. As shown in Figure 10, the weight at the most recent epoch is in the best.pt file, whereas the weight at the most accurate epoch is in the last.pt file. Both files are the same 14.5MB in size.

3.4. Training Results

To get the result, the Yolov5 model is used to identify and find signed Afaan Oromo in the dataset. We quantitatively analyzed our model using recall, precision, and mAP in order to assess the effectiveness of the suggested model. The results of the trained model for each class of accuracy are shown in the table below.

Table 1. *Model summary of prediction for each class.*

No	Sign detection	Precision (%)	Recall (%)	mAP 0.5%)	(mAP 0.5:0.95%)
	All	0.907	0.929	0.932	0.711
	A	0.979	1	0.995	0.804
	Abbaa Manaa	0.986	1	0.995	0.636
	Abbaa	0.904	0.929	0.936	0.641
	Adurree	0.952	1	0.995	0.812
	Afur	0.93	1	0.995	0.901
	Akkam	0.685	0.7	0.635	0.401
	Amma	0.895	1	0.995	0.729
	B	0.979	1	0.995	0.862
	Bicuu	0.985	1	0.995	0.787
	Bilbila	0.985	0.955	0.986	0.639
	Bishaan	0.977	1	0.995	0.919
	Booda	0.991	1	0.995	0.722
	Boor	0.871	0.967	0.971	0.635
	C	0.981	1	0.995	0.862
	D	0.976	1	0.995	0.846
	Dallansuu	0.987	1	0.995	0.715

No	Sign detection	Precision (%)	Recall (%)	mAP 0.5%)	(mAP 0.5:0.95%)
	Dansaa	0.978	0.882	0.912	0.718
	Dheeraa	0.883	1	0.995	0.721
	Dhukkubsataa	0.826	0.761	0.876	0.563
	Dibaabee	1	0.991	0.995	0.888
	Dulloomaa	0.464	0.551	0.446	0.285
	Duwwaa	1	0.842	0.96	0.738
	E	0.977	1	0.995	0.834
	F	0.975	1	0.995	0.744
	Fagoo	0.966	1	0.995	0.832
	Funyaan	0.881	1	0.995	0.895
	Furdaa	0.908	1	0.968	0.871
	Furtuu	0.686	0.764	0.67	0.566
	G	0.981	1	0.995	0.739
	Gabaabaa	0.846	1	0.978	0.841
	Gadduu	0.987	1	0.995	0.904
	Gubbaa	0.986	1	0.995	0.778
	Guutuu	0.973	1	0.995	0.803
	H	0.905	1	0.946	0.689
	Haadha Warraa	0.589	0.6	0.628	0.486
	Haadha	0.93	0.891	0.913	0.584
	Har'a	0.98	1	0.995	0.818
	Hiriyyaa	0.987	1	0.995	0.749
	I	0.989	1	0.995	0.717
	Ija	0.73	0.731	0.759	0.546

3.5. Experiments

Images, webcam footage, and videos are used during testing. We used those weights for inference along with a conf. parameter indicating model confidence and an inference source. A device's camera port, a directory of photos, individual images, and video files can all be accepted as sources.

3.5.1. Testing with Images

In this, we used 711 photos to make up the entire testing dataset for this model. Any image's sign pattern can be recognized using the training weights. A bounding box is drawn to cover the object and reflect the likelihood that it is a sign if the existence of a sign is recognized. The model must be trained in order to recognize signs using trained weights. The "python detect.py" file will be compiled using the command given in Figure 11, and it rebuilds the architecture used in the training. With a 93 percent accuracy rate, trained weights will be utilized to anticipate and limit boxes for them.

```
In [20]: # when we ran this, we saw .007 second inference time. That is 140 FPS on a TESLA P100!
         # use the best weights!
         %cd /content/yolov5/
         !python detect.py --weights runs/train/yolov5s_results/weights/best.pt --img 416 --conf 0.5 --source ../test/images

/content/yolov5
detect: weights=['runs/train/yolov5s_results/weights/best.pt'], source=../test/images, data=data/coco128.yaml, imgsz=[416, 41
6], conf_thres=0.5, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=False,
nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp,
exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5 🚀 v6.1-177-gd059d1d torch 1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
custom_YOLOv5s summary: 232 layers, 7486551 parameters, 0 gradients, 17.5 GFLOPs
image 1/711 /content/test/images/10-ale--11-_jpg.rf.e9ec95eb5e5fefac078c229528333464.jpg: 416x416 1 Kudhan, Done. (0.011s)
image 2/711 /content/test/images/10-ale--30-_jpg.rf.a8a8b95f6498cc813e23d970a96ec9d9.jpg: 416x416 1 Kudhan, Done. (0.011s)
image 3/711 /content/test/images/10-ale--33-_jpg.rf.c244abcffe72dbe53467fa2aada73b22.jpg: 416x416 1 Kudhan, Done. (0.011s)
image 4/711 /content/test/images/10-ale--40-_jpg.rf.22e169e23ba57696f6675f67d26f38a3.jpg: 416x416 1 Kudhan, Done. (0.011s)
image 5/711 /content/test/images/10-ale--47-_jpg.rf.4cf7c54ad3e95029162165dd1062075f.jpg: 416x416 1 Kudhan, Done. (0.011s)
image 6/711 /content/test/images/10-ale--49-_jpg.rf.ed70eb7f63a65d367eaad1ff17793eb2.jpg: 416x416 1 Kudhan, Done. (0.011s)
image 7/711 /content/test/images/10-ale--51-_jpg.rf.465cfea1e3ba4c4ca8220213d6756189.jpg: 416x416 1 Kudhan, Done. (0.012s)
image 8/711 /content/test/images/10-ale--75-_jpg.rf.4aaab3ded976d2beb58347a3de99627e.jpg: 416x416 1 Kudhan, Done. (0.012s)
image 9/711 /content/test/images/10-ale--77-_jpg.rf.6f1f2f23be161beb097e98bd582724a6.jpg: 416x416 1 Kudhan, Done. (0.012s)
image 10/711 /content/test/images/10-dab--10-_jpg.rf.3eae6243632884ad6bb78aba0bfd711b.jpg: 416x416 1 Kudhan, Done. (0.011s)
```

Figure 12. *Detects signs with trained weight.*

The estimated bounding boxes that surround the items will be drawn into the image after detection is finished. They were kept in the same folder as the training phase's results.

If all the steps were followed, the command was output as follows, and training could then begin. We can learn how the model is run via mAP@.5. Once training has started, the Yolov5 training pipeline inputs test image ground truth and prediction results in the "run" folder as displayed below.

```
In [22]: import glob
         from IPython.display import Image, display

         for imageName in glob.glob('/content/yolov5/runs/detect/exp2/*.jpg'): #assuming JPG
             display(Image(filename=imageName))
             print("\n")
```

Figure 13. *Load predicted images as array format for visualization.*

Figure 14. *Detected labels from test images.*

3.5.2. Testing with Webcam Feed

We used a CPU to recognize signs in real time while testing with a webcam. Computer speed is necessary to forecast a sign within a second. Lack of a GPU, however, makes it difficult to evaluate it in real-time. To evaluate its performance, we ran the following command to see how well it could recognize signs using the web camera. The results are shown in the picture below.

```
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\user\Desktop\yolov5>python detect.py --weights runs/train/yolov5s_results/weights/best.pt --source 0
```

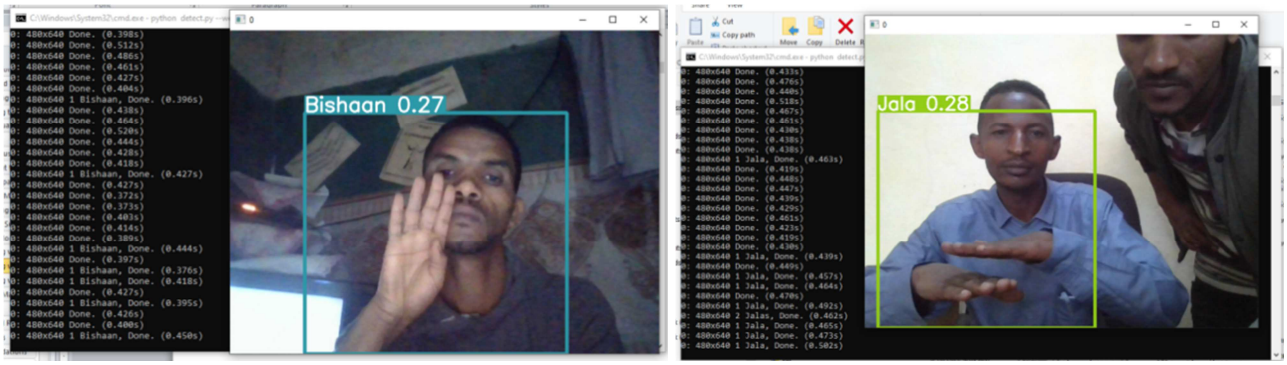

Figure 15. *Tested with a webcam.*

3.5.3. Testing with Video

The data was processed on several movies that the author had shot, separating it from the data from the videos that were trained in situations comparable to the testing conditions. The dimensions of testing videos are 480 x 640. The movie consists of two or more signs that are successively displayed, generating a string of letters that correspond to the video's goal texts. With the highest degree of certainty, only one sign will be detected overall. When signs are found in a sequence that approach the predetermined limit, they are regarded as recognized signs and are compared to the given ground truth.

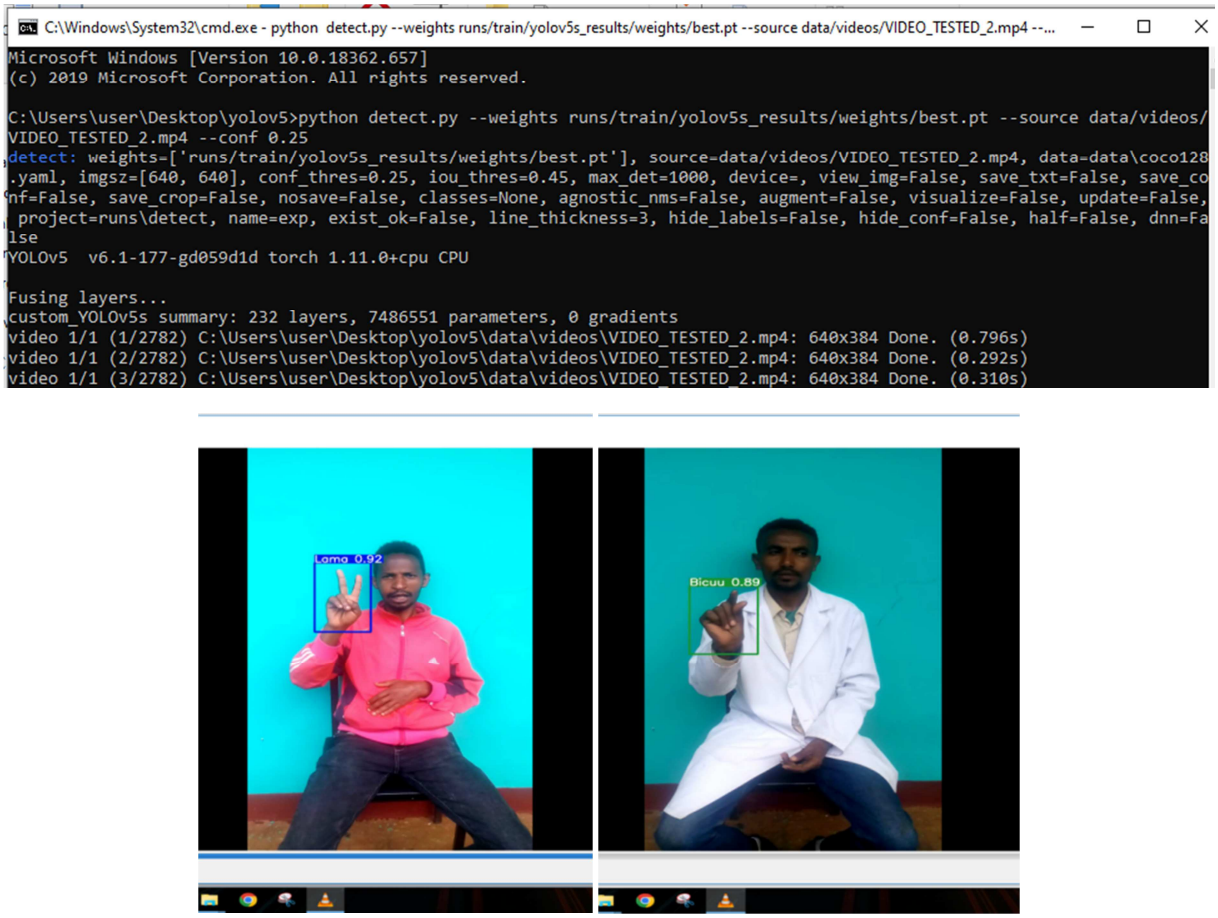

Figure 16. *Predictions from videos dataset.*

3.6. Evaluation Metric by Using mAP

A statistic known as Average Precision (AP) and mean average precision is used to assess the effectiveness of the sign language detection and localization (mAP) [31]. Over recall levels ranging from 0 to 1, the average accuracy (AP) values (mean) are calculated. Confusion Matrix, Intersection over Union (IoU), Recall, and Precision measurements are the foundation of the mAP formula. We require four attributes in order to generate a confusion matrix [5, 32]:

If IoU ≥0.5, the detection signal should be classified as True Positive (TP), If IoU <0.5, after, it is a false detection and is labeled as such False Positive (FP) and When the image contains ground truth but the model is unable to identify the object, we categorize it as a False Negative (FN).

Intersection over Union (IOU): This is a situation in which the model's predicted bounding box of an object and its overlap with the object's original bounding box in the picture determine whether or not detection is accurate (ground truth) [26]. Another name for this is Intersection over Union. It is defined as the intersection of the expected and actual bounding boxes divided by the sum of those two. If IoU > threshold, a forecast is deemed True Positive; if IoU threshold, it is deemed False Positive.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{1}$$

Precision is the model's level of accuracy in identifying just pertinent things. It measures the proportion of TPs to all detections the model has produced.

$$P = \frac{TP}{TP+FP} = \frac{TP}{All\ detections} = 0.90 \tag{2}$$

Recall gauges a model's capacity to identify every TPs proposition among all ground truths.

$$R = \frac{TP}{TP+FN} = \frac{TP}{All\ ground\ truths} = 0.92 \tag{3}$$

Precision Recall Curve: Based on the IoU, we are given a confidence score and a TP or FP designation for each prediction [33]. The TP and FP are then added up after the results are sorted by confidence score. We determine the Precision and Recall for each prediction using the summed TP and FP. A Precision and Recall Curve can now be used to represent this result.

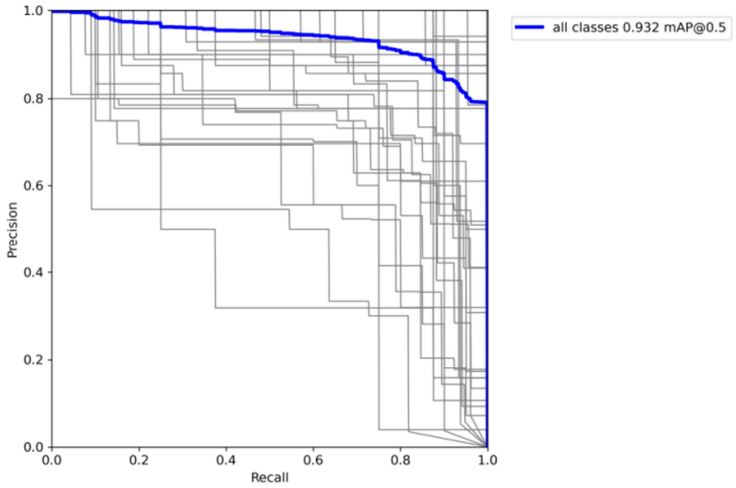

Figure 17. *PR curve.*

Mean Average Precision (mAP): The AP, which has a fixed IoU (IoU>0.5), is the average of the maximum precisions at various recall settings.

$$\text{mAP } 0.5\% = 0.995+0.995+0.936+0.995+0.995\ldots.90n = 83.892/90 = 0.932.$$

A matrix called mAP is used to evaluate the precision of body detectors. Average AP score across all classes.

$$(\text{mAP } 0.5{:}0.95\%) = 0.636+0.641+0.812+0.901+0.401+0.729+0\ldots n90 = 63.988/90 = 0.\ 711$$

4. Conclusions

The communication gap between the deaf and hearing persons has been brought to light by the current research, which has also developed an implementation of Signed language to AO texts. Research on sign language translation is ongoing. The fundamental concept of speech-impaired people communicating by sign language is explained, as well as the issues they face in the absence of an interpreter. Therefore, the goal of this project is to build a sign language translator that will enable them to easily converse with non-signers. In the absence of a sign language interpreter, they can also communicate. CNN and YOLO concepts for object detection and hand gesture classification are used in sign language translation. The model also has several flaws. While being detected, some words and numerals were unclassified, and some letters did not receive predictions. The letters that were mistakenly predicted are "0" forecasted as "O" and dynamic words are not more predicted, as can be seen from the confusion matrix. Isaan, Ishee, Isa, Dheera, Abba, Haadha, J, and Z are a few examples. Recall declines monotonically whereas precision fluctuates as the confidence score rises, but for this all classes rise. The new sign gesture translation algorithm can recognize gestures from videos in real-time with an accuracy of 93%. We correctly anticipated all of the chosen numerals, alphabets, and words. The widespread consensus is that "Computer vision can and must be employed in designing a step towards expanding availability of educational materials for our deaf community."

Data Availability

The Sign language image, video and implementation code we congregated is with authorization of the apprehensive part.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

I would also like to forward my deepest respect and appreciation to my advisor, Kasahun Abdisa (PhD) and my co-advisor Mr. Etana Fikadu (PhD) for their ultimate help and guidance in the successful completion of this paper. Regarding the dataset sign language teachers found at Nekemte Church of Christ KG and deaf school, Mr. Isayas Wandimu, Ms. Kasech Gabisa, Mr. Getu Tasfaye, and Mr. Dhaba Kamiso. Next, I would like to thank Ms. Ayelech Tesema (PhD) from Sebeta Special Needs Education Teachers college Instructor and Ms. Kasech Tadesse (PhD) from Wollega University, department of behavioral science and special needs instructor and Mr. Alamayow Etana from Wollega University.

References

[1] A. Tsegay and K. Raimond, "Offline candidate hand gesture selection and trajectory determination for continuous Ethiopian sign language," *J. Theor. Appl. Inf. Technol.*, vol. 36, no. 1, pp. 145–153, 2012.

[2] Ibrahim Bedane, "The Origin of Afaan Oromo : Mother Language," *Double Blind Peer Rev. Int. Res. J.*, vol. 15, no. 12, 2015.

[3] D. Bijiga, "The Development of Oromo Writing System School of European Culture and Languages," *Http://Kar.Kent.Ac.Uk*, pp. 1–295, 2015, [Online]. Available: https://kar.kent.ac.uk/52387/

[4] V. Adithya and R. Rajesh, "A Deep Convolutional Neural Network Approach for Static Hand Gesture Recognition," *Procedia Comput. Sci.*, vol. 171, no. 2019, pp. 2353–2361, 2020, doi: 10.1016/j.procs.2020.04.255.

[5] M. Nanda, "You Only Gesture Once (Yougo): American Sign Language Translation Using Yolov3," no. May, 2020.

[6] K. Brady *et al.*, "American Sign Language Recognition and Translation Feasibility Study. | National Technical Reports Library - NTIS," pp. 3–76, 2018, [Online]. Available: https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/AD1098892.xhtml

[7] H. D. Patel and A. Saluja, "Sign Language Recognition and Translator Application," *Int. Res. J. Eng. Technol.*, pp. 652–658, 2021, [Online]. Available: www.irjet.net

[8] A. Halder and A. Tayade, "Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network,"," *Int. J. Res. Publ. Rev.*, vol. 8, no. 2, pp. 9–17, 2021.

[9] S. Krishnamurthi and M. Indiramma, "Sign Language Translator Using Deep Learning Techniques," *2021 4th Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2021*, 2021, doi: 10.1109/ICECCT52121.2021.9616795.

[10] K. Yin, M. Alikhani, A. Moryossef, J. Hochgesang, and Y. Goldberg, "Including Signed Languages in Natural Language Processing (Extended Abstract)," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 5369–5373, 2022, doi: 10.24963/ijcai.2022/753.

[11] W. Chen, Z. Liqiang, Y. Tianpeng, J. Tao, J. Yijing, and L. Zhihao, "Research on the state detection of the secondary panel of the switchgear based on the YOLOv5 network model," *J. Phys. Conf. Ser.*, vol. 1994, no. 1, 2021, doi: 10.1088/1742-6596/1994/1/012030.

[12] S. S. Sharma *et al.*, "Effective Face Detector Based on YOLOv5 and Superresolution Reconstruction," *Comput. Math. Methods Med.*, vol. 2021, no. June, pp. 1–9, 2021, doi: 10.1155/2021/7748350.

[13] L. Jiang, H. Liu, H. Zhu, and G. Zhang, "Improved YOLO v5 with balanced feature pyramid and attention module for traffic sign detection," *MATEC Web Conf.*, vol. 355, p. 03023, 2022, doi: 10.1051/matecconf/202235503023.

[14] S. M. Ali, "Comparative Analysis of YOLOv3, YOLOv4 and YOLOv5 for Sign Language Detection," *Int. J. Adv. Res. Innov. Ideas Educ.*, vol. 7, no. 4, pp. 2393–2398, 2021, [Online]. Available: http://ijariie.com/AdminUploadPdf/Comparative_Analysis_of_YOLOv3__YOLOv4_and_YOLOv5_for_Sign_Language_Detection_ijariie15253.pdf

[15] D. Thuan, "Evolution of Yolo Algorithm and Yolov5: the State-of-the-Art Object Detection Algorithm," p. 61, 2021.

[16] T. F. Dima and M. E. Ahmed, "Using YOLOv5 Algorithm to Detect and Recognize American Sign Language," *2021 Int. Conf. Inf. Technol. ICIT 2021 - Proc.*, no. December, pp. 603–607, 2021, doi: 10.1109/ICIT52682.2021.9491672.

[17] Q. Xu, Z. Zhu, H. Ge, Z. Zhang, and X. Zang, "Effective Face Detector Based on YOLOv5 and Superresolution Reconstruction," *Comput. Math. Methods Med.*, vol. 2021, 2021, doi: 10.1155/2021/7748350.

[18] J. H. Kim, N. Kim, Y. W. Park, and C. S. Won, "Object Detection and Classification Based on YOLO-V5 with Improved Maritime Dataset," *J. Mar. Sci. Eng.*, vol. 10, no. 3, 2022, doi: 10.3390/jmse10030377.

[19] W. Wu *et al.*, "Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image," *PLoS One*, vol. 16, no. 10 October, pp. 1–15, 2021, doi: 10.1371/journal.pone.0259283.

[20] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[21] W. Jia *et al.*, "Real-time automatic helmet detection of motorcyclists in urban traffic using improved YOLOv5 detector," *IET Image Process.*, vol. 15, no. 14, pp. 3623–3637, 2021, doi: 10.1049/ipr2.12295.

[22] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, 2020, doi: 10.1109/TPAMI.2018.2858826.

[23] A. Ananthanarayan, "Hand Detection and Body Language Recognition Using YOLO," no. May, 2021, [Online]. Available: https://hdl.handle.net/1969.1/188381

[24] M. Mulugeta, "Development of an Amharic Speech to Ethiopian Sign Language Translation System.," *Bahirdar universty, Ethiop.*, 2016.

[25] D. Solanki, "OBJECT DETECTION AND CLASSIFICATION THROUGH DEEP LEARNING APPROACHES," vol. 5, no. 9, pp. 230–235, 2018.

[26] S. L. Kuna, "Real Time Object Detection and Tracking using Deep Learning," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 8, no. 6, pp. 58–64, 2020, doi: 10.22214/ijraset.2020.6010.

[27] E. Martinez-Martin and F. Morillas-Espejo, "Deep Learning Techniques for Spanish Sign Language Interpretation," *Comput. Intell. Neurosci.*, vol. 2021, no. i, 2021, doi: 10.1155/2021/5532580.

[28] M. Rivera-Acosta, J. M. Ruiz-Varela, S. Ortega-Cisneros, J. Rivera, R. Parra-Michel, and P. Mejia-Alvarez, "Spelling correction real-time american sign language alphabet translation system based on yolo network and LSTM," *Electron.*, vol. 10, no. 9, 2021, doi: 10.3390/electronics10091035.

[29] S. Daniels, N. Suciati, and C. Fathichah, "Indonesian Sign Language Recognition using YOLO Method," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1077, no. 1, p. 012029, 2021, doi: 10.1088/1757-899x/1077/1/012029.

[30] I. Cardoza, J. P. García-Vázquez, A. Díaz-Ramírez, and V. Quintero-Rosas, "Convolutional Neural Networks Hyperparameter Tunning for Classifying Firearms on Images," *Appl. Artif. Intell.*, vol. 36, no. 1, pp. 1–23, 2022, doi: 10.1080/08839514.2022.2058165.

[31] K. Guo, C. He, M. Yang, and S. Wang, "A pavement distresses identification method optimized for YOLOv5s," *Sci. Rep.*, vol. 12, no. 1, pp. 1–15, 2022, doi: 10.1038/s41598-022-07527-3.

[32] D. K. Singh, "3D-CNN based Dynamic Gesture Recognition for Indian Sign Language Modeling," *Procedia CIRP*, vol. 189, pp. 76–83, 2021, doi: 10.1016/j.procs.2021.05.071.

[33] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, [Online]. Available: http://arxiv.org/abs/1804.02767