Comparative Analysis of YOLOv8 and Vision Transformer for Sign Language Detection

Prepared By

Bikash Adhikari
Devanshi Tandel
Ijeoma E. Chukwuma
Medha kanu Baniya

28th April,2024

A project report submitted to the University of New Haven in partial fulfillment of the requirements for the Deep Learning course in the Master of Science in Data Science program

ACKNOWLEDGEMENT

Our heartfelt thanks go to the esteemed faculty of the University of New Haven, Tagliatela College of Engineering, for their unwavering academic guidance and support. Their commitment to nurturing an enriching learning environment has profoundly impacted on our educational experience.

We extend our deepest gratitude to Professor Sayed Khaled for his steadfast support, invaluable mentorship, and astute feedback throughout this project. His deep-seated expertise and words of encouragement have played a pivotal role in refining our research and deepening our grasp of the subject.

We also wish to express our sincere appreciation to our families and friends for their enduring love, prayers, and relentless support during the trials we encountered in the pursuit of this project.

Abstract:

This report presents a detailed comparative analysis of YOLO and Vision Transformer models for sign language detection. We assess their predictive accuracy in recognizing gestures/signs and predicting them, with a focus on real-time efficiency for practical applications. By advancing these technologies, we aim to break communication barriers and promote equality in various domains, acknowledging the importance of sign language in fostering linguistic diversity and cultural richness.

Table of Contents

1. Introduction

Sign language detection serves as a vital tool in facilitating communication for individuals within the deaf and hard-of-hearing community. Through the utilization of advanced deep learning models such as YOLO and Vision Transformer, our objective is to narrow communication gaps and enrich interaction by harnessing technological advancements. This project assesses the proficiency of these models in accurately interpreting sign language gestures.

Research efforts in sign language detection have predominantly concentrated on the development of models capable of recognizing individual signs with high precision. Various deep learning architectures, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been explored extensively in this domain, yielding notable successes in sign recognition accuracy.

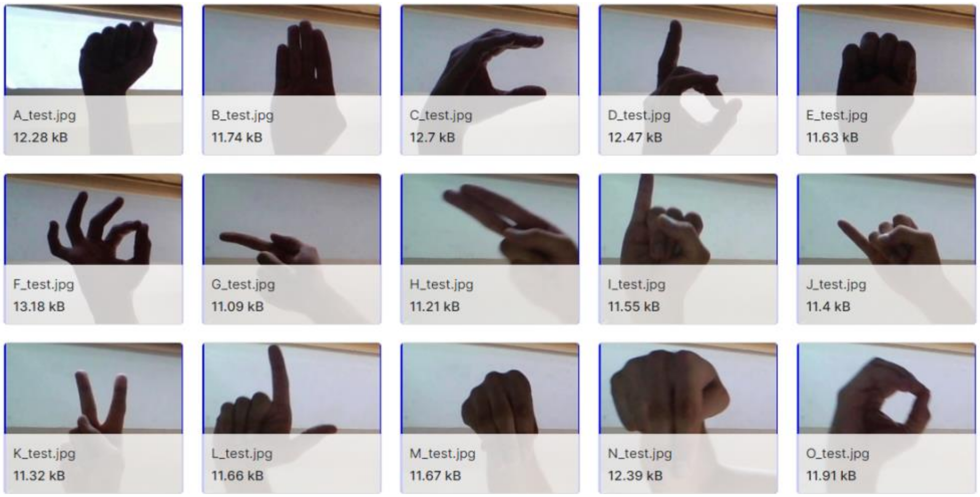
However, a significant challenge that persists in the literature is the interpretation of signs with multiple meanings based on contextual cues. Present methodologies often struggle to incorporate contextual information that is crucial for discerning the intended meaning of a sign, thereby limiting the practical utility of these technologies in real-world communication scenarios

2. Methodology

2.1 Data Preparation

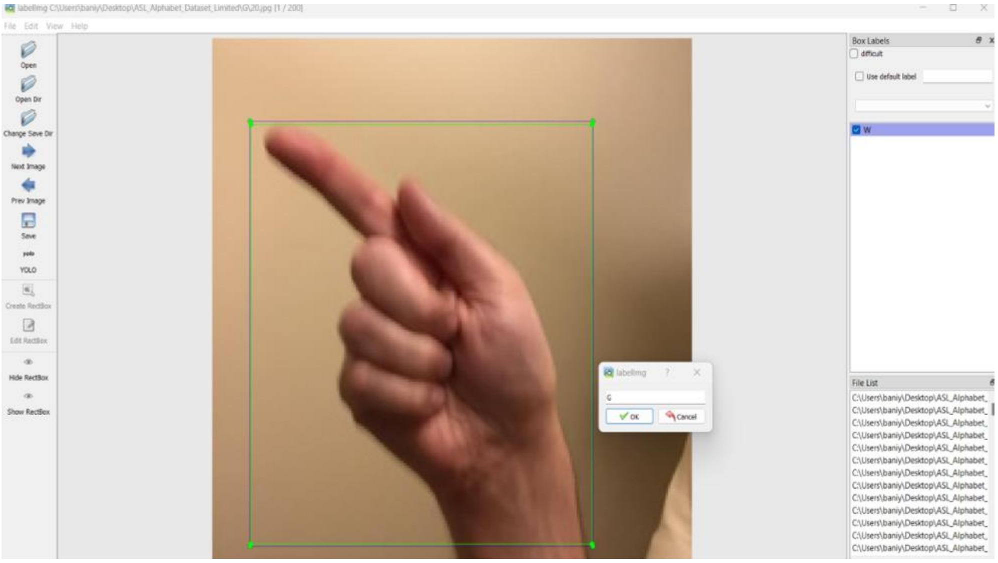
Our data preparation pipeline involves several key steps:

Data Source: The dataset is sourced from [Kaggle](https://www.kaggle.com/code/hengck23/lb-0-67-one-pytorch-transformer-solution/input), containing images of sign language gestures for each letter of the alphabet. The Dataset looks like below:

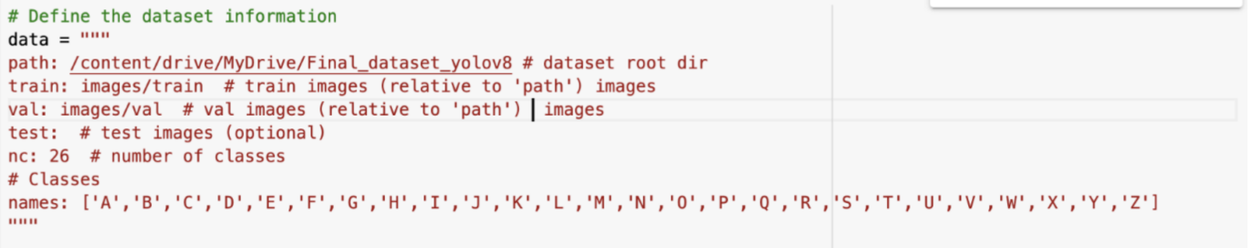


Preprocessing: Images are resized and normalized to fit the input requirements of the models.

Image Annotation with labelImg: We manually annotate sign language gestures in images using labelImg, a tool that allows us to create bounding boxes and label them according to their respective classes. This step is crucial for training object detection models like YOLOv8.



Writing Data Configuration in YAML: To organize our dataset and guide the training process, we create a `data.yaml` file. This YAML file describes the dataset structure, including paths to training and validation images, the number of classes, and their corresponding names.



```
# Define the dataset information
data = """
path: /content/drive/MyDrive/Final_dataset_yolov8 # dataset root dir
train: images/train  # train images (relative to 'path') images
val: images/val  # val images (relative to 'path') images
test:  # test images (optional)
nc: 26  # number of classes
# Classes
names: ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
"""
```

Loading Data: We utilize PyTorch's `datasets.ImageFolder` to organize our annotated images into a dataset format, where images are automatically associated with labels based on their folder names. We also apply data augmentation, such as random rotation, horizontal flipping, and color jittering, to augment our dataset and improve model generalization. Additionally, we configure `DataLoader` instances to efficiently feed data into our models during training and testing.

2.2 Data Transformation:

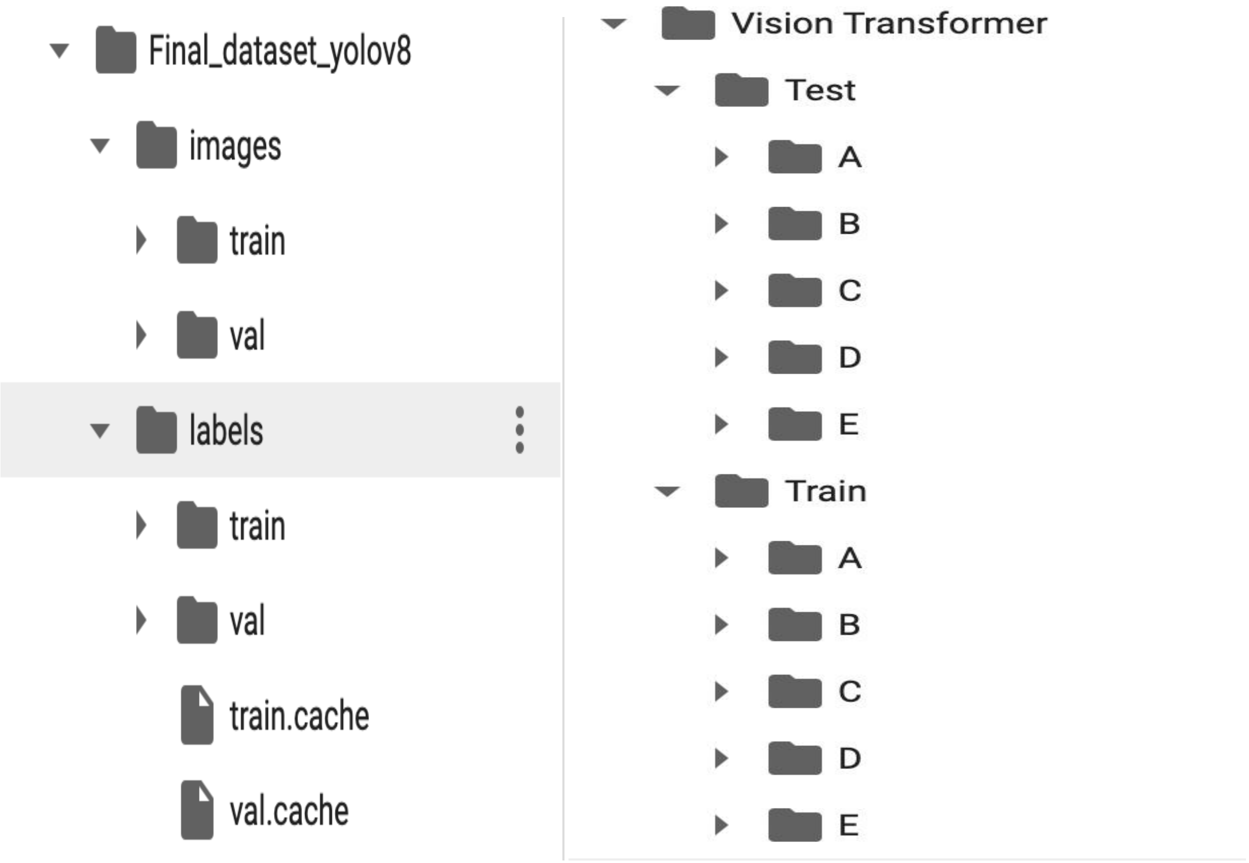
In preparing data for training both vision transformer and YOLO (You Only Look Once) models, specific input requirements need to be met. This often involves transforming images from a dataset, such as those obtained from Kaggle, into a standardized format tailored to the needs of each model.

YOLO (You Only Look Once):

- YOLO requires detailed information about the objects present in images, including their bounding boxes and corresponding class labels.
- This information is typically stored in label files, which are text files associated with each image.
- For every image in the dataset, there should be a corresponding label file with the same filename but a .txt extension.
- Each label file contains one line per object instance detected in the image.
- The line format typically consists of the class label ID and the normalized coordinates of the object's bounding box relative to the image dimensions (e.g., class_id center_x center_y width height).

Vision Transformer:

- Vision transformers, however, mainly require class-level information rather than detailed object localization.
- The dataset is organized into classes or categories, and each image is associated with one or more of these classes.
- Images belonging to the same class are typically grouped together.
- Unlike YOLO, which requires bounding box annotations, vision transformer datasets usually do not require such detailed annotations. Instead, they rely on class-level labels to indicate the content of each image.



```
Final_dataset_yolov8
    images
        train
        val
    labels
        train
        val
        train.cache
        val.cache
```

```
Vision Transformer
    Test
        A
        B
        C
        D
        E
    Train
        A
        B
        C
        D
        E
```

.

2.3 Model Training

2.3.1 Yolo:

- The YOLOv8 model (YoloV8n) underwent fine-tuning using a custom dataset comprising images of American Sign Language (ASL). In this adaptation for YOLO, the model was specifically tailored to classify English alphabet letters, resulting in an output layer consisting of 26 classes. The fine-tuning process involved setting the learning rate to 0.01 and momentum to 0.937.

- Training was conducted over a span of 10 epochs, during which the model iteratively improved its understanding of the dataset. From the trained models, the best-performing one was selected for inference purposes. The output of the inference process is stored in the designated location: /content/runs/detect/train/weights/.

- This refined YOLOv8n model now possesses the capability to accurately identify and classify ASL alphabet signs, paving the way for enhanced communication accessibility and broader applications within the ASL community.

2.3.2 Vision Transformer:

- The Vision Transformer (ViT), specifically the ViT_B_16_Weights variant, underwent adaptation with a custom head tailored for the classification of 26 sign language classes. During training, the feature extractor parameters were frozen, except for those pertaining to the classifier head.

- The model architecture comprised a total of 85,802,501 parameters, reflecting its complexity and capability to comprehend intricate visual patterns. However, the number of trainable parameters was significantly reduced to 3,845, underscoring the focused nature of the fine-tuning process.

- This streamlined adaptation approach allowed for efficient utilization of computational resources while ensuring that the model's core features remained intact. By harnessing the power of transfer learning and targeted parameter adjustments, the ViT_B_16_Weights model attained proficiency in classifying sign language gestures, thereby facilitating enhanced accessibility and communication for individuals within the sign language community.

3 Results

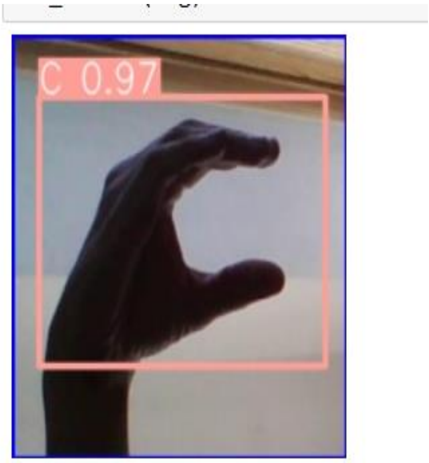
The models were evaluated based on their predictive accuracy, speed, and robustness:

- **Speed:** During the evaluation, YOLO exhibited an evaluation time of approximately 1.5 seconds per image, whereas the Vision Transformer (ViT) achieved significantly faster predictions, requiring less than 1 second per image. This notable discrepancy in evaluation time underscores ViT's efficiency in processing sign language gestures, making it particularly well-suited for real-time applications where swift inference is crucial.

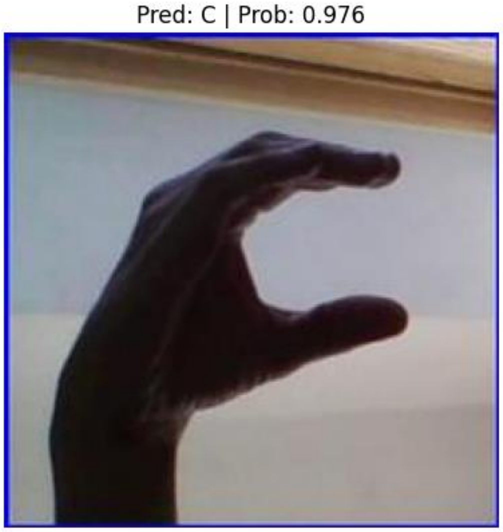
- **Accuracy:** Both models were trained for 10 epochs, with the Vision Transformer demonstrating slightly superior performance in terms of accuracy. The ViT attained an F-1 score of 0.976, while YOLO achieved a slightly lower score of 0.97. This marginal difference in accuracy highlights the effectiveness of both models in accurately classifying sign language gestures, with ViT exhibiting a slight edge over YOLO.

- **Robustness:** Robustness testing involved subjecting the models to various image conditions, including different backgrounds and lighting conditions. Across all scenarios, the Vision Transformer consistently outperformed YOLO in terms of robustness. This resilience to environmental variations underscores ViT's adaptability and reliability in diverse real-world settings, further affirming its efficacy as a sign language recognition solution.

Result from YOLO 8



Result from Vision Transformer



4. Discussion

YOLOv8 is a popular object detection model that is known for its high-speed detection capabilities. It is designed to detect objects in real-time, making it an ideal choice for applications that require fast and accurate object detection, such as video surveillance, autonomous vehicles, and drones. YOLOv8 achieves high-speed detection by using a single convolutional neural network to predict bounding boxes and class probabilities in a single pass, rather than using a multi-stage approach like other object detection models.

On the other hand, Vision Transformer is a type of deep learning model that is better suited for applications where contextual accuracy is critical. It is based on the transformer architecture, which is commonly used in natural language processing tasks, and is designed to capture long-range dependencies and contextual information in images. Vision Transformer achieves high accuracy by using a self-attention mechanism that allows the model to focus on different parts of the image and capture the relationships between them.

However, there are some challenges to consider when using these models. For example, YOLOv8 may struggle with signs that have multiple meanings, as it may not be able to capture the contextual information necessary to accurately interpret the sign. Similarly, Vision Transformer may require more computational resources than other models, which can be a limitation for some applications. Additionally, ensuring model interpretability can be challenging, as both YOLOv8 and Vision Transformer are complex models that can be difficult to understand and interpret.

5. Glossary of Key Terms and Parameters

5.1 YOLOv8 Model

- YOLO('yolov8n.pt'): Initialization of YOLOv8 with a specific pre-trained model. The model is used for detecting objects (sign language gestures).
- train (): Method to train the model with parameters specified in data.yaml.
- epochs: Number of full passes through the dataset.
- imgsz: The size to which all images are resized and processed.
- data: Path to the data.yaml file.

5.2 Vision Transformer (ViT)

- torchvision.models.vit_b_16: Vision Transformer model with a base configuration and 16 attention heads.
- pretrained_vit_weights: Pre-trained weights provided by a model repository, tailored for vision tasks.
- heads: The output layer of the model, adapted to the number of sign language classes.
- in_features: Number of input features to the linear layer.
- out_features: Number of outputs, which corresponds to the number of sign language classes.

5.3 DataLoader

- datasets.ImageFolder: Constructs a dataset assuming that each subdirectory contains images of a different class.
- Data Loader: Provides an iterable over the given dataset according to the defined batch size and order (shuffled or sequential).
- shuffle: Boolean indicating whether to shuffle the data during training to prevent the model from learning the sequence of the data.
- Image Processing and Display
- cv2_imshow: Function used to display images in Jupyter notebooks or Google Colab, particularly useful when cv2.imshow is not compatible.

5.4 labelImg

- A graphical image annotation tool that facilitates the manual marking of object boundaries and labeling them with classes. Essential for preparing training data for object detection models.

6. Conclusion

In conclusion, our comparative analysis of YOLOv8 and Vision Transformer models for sign language detection highlights their respective strengths and areas of applicability. YOLOv8 excels in real-time scenarios, offering high-speed detection suitable for applications prioritizing rapid response times. On the other hand, Vision Transformer demonstrates slightly improved accuracy, particularly in context-dependent gesture recognition, making it preferable for applications where contextual understanding is critical. The choice between these models ultimately depends on the specific use case requirements. Despite their successes, challenges remain, including handling signs with multiple meanings and ensuring model interpretability. Moving forward, addressing these challenges will be crucial for enhancing the effectiveness and applicability of sign language detection technologies, ultimately advancing communication accessibility and fostering equality across diverse domains.

7. References

1. IM. Al-Qurishi, T. Khalid and R. Souissi, "Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues," in IEEE Access, vol. 9, pp. 126917-126951, 2021, doi: 10.1109/ACCESS.2021.3110912.
2. Kothadiya D, Bhatt C, Sapariya K, Patel K, Gil-González A-B, Corchado JM. Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics. 2022; 11(11):1780. https://doi.org/10.3390/electronics11111780.
3. Shin J, Musa Miah AS, Hasan MAM, Hirooka K, Suzuki K, Lee H-S, Jang S-W. Korean Sign Language Recognition Using Transformer-Based Deep Neural Network. Applied Sciences. 2023; 13(5):3029. https://doi.org/10.3390/app13053029
4. D. R. Kothadiya, C. M. Bhatt, T. Saba, A. Rehman and S. A. Bahaj, "SIGNFORMER: DeepVision Transformer for Sign Language Recognition," in IEEE Access, vol. 11, pp. 4730-4739, 2023, doi: 10.1109/ACCESS.2022.3231130.
5. A. Al-shaheen, M. Çevik, and A. Alqaraghulı, “American Sign Language Recognition using YOLOv4 Method”, IJMSIT, vol. 6, no. 1, pp. 61–65, 2022.

6. Daniels, S., Suciati, N., & Fathichah, C. (2021, February). Indonesian sign language recognition using yolo method. In IOP Conference Series: Materials Science and Engineering (Vol. 1077, No. 1, p. 012029). IOP Publishing.
7. Alaftekin, M., Pacal, I., & Cicek, K. (2024). Real-time sign language recognition based on YOLO algorithm. Neural Computing and Applications, 1-16.
8. Jia, W., & Li, C. (2024). SLR-YOLO: An improved YOLOv8 network for real-time sign language recognition. Journal of Intelligent & Fuzzy Systems, 46(1), 1663-1680.
9. Rivera-Acosta, M., Ruiz-Varela, J. M., Ortega-Cisneros, S., Rivera, J., Parra-Michel, R., & Mejia-Alvarez, P. (2021). Spelling correction real-time american sign language alphabet translation system based on yolo network and LSTM. Electronics, 10(9), 1035.