VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

SIGN LANGUAGE TRANSLATION MODEL FOR VIETNAMESE

By
Nguyen Duc Toan

A thesis submitted to the School of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Computer Science

Ho Chi Minh City, Vietnam
2025

SIGN LANGUAGE TRANSLATION MODEL FOR VIETNAMESE

APPROVED BY:

_____,
Committee name here

_____
Committee name here

_____
Committee name here

_____
Committee name here

_____
Committee name here

THESIS COMMITTEE

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Le Thi Ngoc Hanh, for all her help and support during my research. She gave me very good advice and feedback that made my research paper much better. Dr. Hanh was always patient with me and helped me understand difficult concepts. Her suggestions and encouragement helped me complete this work successfully. She taught me how to do good research, give comment on my study as soon as possible, answer my questions immediately when I need and always pushed me to do my best work.

I am very grateful to my friends and family for all their support. I could not have done this research without their help and understanding. When I was working long hours on this project, they were always there, stand by me to encourage me and cheer me up. Their belief that I could succeed kept me going when things got difficult. They listened to me when I was worried and celebrated with me when I made progress. Having people who care about me made this whole journey much easier.

I want to give special thanks to my parents, who helped me in so many ways. Their unwavering belief in the importance of education, combined with their generous financial support, has made this academic achievement possible. More than the material assistance, their constant motivation, wise counsel, and emotional support have been the foundation upon which my academic aspirations have been built. My parents always believed in me and reminded me that I could reach my goals. They made many sacrifices to help me succeed in school. When I felt tired or wanted to give up, they were there to remind me why this work was important. Without their love and supp ort, I would not have been able to finish this research.

Finally, I know that this research would not have been possible without all the people I mentioned above. Each person played an important part in helping me complete this work successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Sign language is a visual means of communicating through hand gestures, body movements and facial expressions. It is the language for people with hearing and speaking disabilities to connect their thoughts and ideas to the communities without spoken words. This is one of the most important elements in the daily lives of the impaired community, but it is still not commonly known by other people, which makes it difficult to communicate and make the people with disabilities deprived of basic human rights like healthcare, education and even cannot get a job with minimum wage to cover their life. The reason is simply because of their inability to communicate with others using spoken language or the others have difficulties to understand what they want to express.

Thanks to the development of AI technologies, many Sign Language Detection models have been introduced to revolutionize the way sign language is integrated into daily life, enhancing communication, accessibility, and inclusively. Bridging the gap between the deaf-mute community with the rest of the world, giving the disability community an equal and normal life. Our study aims to connect the communication gap that has been existing in the Vietnamese community for a long time by providing a proposed YOLO model based on the YOLO11 model, the latest iteration in the Ultralytics YOLO series of real-time object detectors, this will play as a real-time solution Vietnamese Sign Language (VSL) translator. The study showed a significant improvement as compared with the other studies in recent years in Vietnam, using the conventional techniques like Support Vector Machine (SVM). After training in 200 epochs with a self created dataset of 20 common Vietnamese words in VSL, our model gives a precision rate of 97.3%, a recall rate of 99.5% and 99.3% mAP score for the mAP@0.5. This shows that this approach has the great capability to accurately detect and classify complex multiple Vietnamese Sign Language. We believe that this model will ensure that everyone has an equal place in Vietnamese society by overcoming the language barrier, and it will assist disabled Vietnamese people interact more effectively with their fellow citizens by resolving the current state of communication gaps.

CHAPTER 1

INTRODUCTION

1.1 Background

Sign languages are a vital means of communication for the Deaf and hard–of-hearing communities, known as a grammatically complete and copious language. According to data from the World Health Organization, "there are around 430 million people suffering from hearing loss around the world in 2024, equal to 5% of the world's population, including 34 million children. It is estimated that this number will exceed 700 million – or 1 in every 10 people – by 2050" (World Health Organization, 2024) [1]. Recognizing and understanding sign language not only brings a connection between the impaired people to the general population, but also plays an important part in many cutting-edge automation systems and applications such as computer-human interactions, robotics, smart homes, etc. In general, sign language is described by four basic components: hand shape, location relative to the body, hand movement, and palm orientation. However, they are not used as international languages because of the different expressions of signs and grammatical structure among countries, regions. In this paper, I focus on recognizing Vietnamese Sign Language (VSL), which holds a unique position as the primary mode of communication for the Deaf and Muted community in Vietnam. Rooted in the country's cultural and linguistic context, VSL possesses its distinct grammar, syntax, and vocabulary, making it an integral part of the linguistic heritage of Vietnam. It shares some features with other sign languages but remains unique in its structure and usage. Despite its significance and crucial in Vietnamese society, resources and tools for VSL recognition remain limited, highlighting the need for research and technological advancements to support its development and accessibility.

1.2 Problem Statement

In recent years, there has been growing interest in the field of sign language recognition using deep learning. The research direction has been divided into two main approaches of sign language detection: sensor-based and vision-based. In the approach based on sensors and actuators, the signers usually wear sensor gloves when describing words and making hand signs, this approach simplifies the preprocessing stage in sign language recognition by removing some outliers from the complex environment and gives a high accuracy in real-time word detections. However, it can be seen that this approach is not practical and not suitable in the real world when signers need to wear a pair of gloves during communication. On the other hand, the vision-based approach includes convolutional neural networks (CNNs), recurrent neural networks (RNNs), or transformer-based models have been introduced to solve the challenges of recognizing hand gestures without any extra equipment attached to the signers. Additionally, vision-based data is affordable and easy to collect compared to sensor-based data. The weakest point of using neutral networks is that the models do not capture enough information about complex motion features. It means that the neutral network models have difficulty handling words which are shown by complicated gestures.

1.3 Scope and Objectives

This research focuses on the application of YOLO11 [2] and propose a lightweight YOLO model based-on the architecture of YOLO11 to recognize hand gestures in Vietnamese Sign Language. YOLO (You Only Look Once) models have demonstrated remarkable performance in real-time object detection tasks, making them well-suited for the dynamic and real-time nature of sign language recognition [3]. And also much research has been done when using YOLO-based models for detecting and translating American Sign Language (ASL), achieving high accuracy when testing on real-time applications. However, limited work has been done to address the unique challenges associated with recognizing VSL. This paper builds on these advancements by leveraging the proposed YOLO model to create a robust and efficient system for recognizing VSL gestures and combine with a Transformer model to create a sentence formation system for recognized words. Through this work, we would like to contribute to the development of accessible communication technologies that empower the Deaf community in Vietnam and beyond. By adapting these state-of-the-art models, we aim to address the challenges posed by the complex and context-dependent gestures in VSL.

The major contributions in the thesis is summarized in the following points:

- Preparing a training VSL dataset.
- Make tweaks on model architecture and loss functions.
- Post processing and building the web application.

1.4 Structure of Thesis

This paper is organized as follows. The analysis of related works is presented in Chapter 2. In Chapter 3, we will describe the overview of methodology including the VSL dataset exploration and preparation, the effect of preprocessing step and data augmentation on the dataset, training configuration, and the training stage. The introduction of our proposed YOLO model and the application of the Transformer model also included in Chapter 3. The experiment results and discussion, the comparison with other recent studies and with the original YOLO11 model, as well as the application created from our proposed model can be found in Chapter 4. Finally, the paper concludes with our findings, summarizing our results and limitations of the research as well as some future works in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

From the last decade there are two main approaches for sign language recognition: sensor-based and vision-based. This chapter explores relevant studies, and techniques introduced in the field, summarizing their results, and highlighting their strengths and limitations.

2.1 Sensor-based Approach

In this approach [4], Bui T. D. and Nguyen L. T. use Microelectronic Mechanical System (MEMS) and the AcceleGlove, which is a wearable computer with super-small electronic circuitry. The sensors in the glove work with a micro-controller attached to the wearer's arm to register the positioning and movement of the arm, fingers, and palm (Hernandez-Rebollar, 2003) [5]. The collected information is turned into data a computer can read. In addition to the five sensors as in the AcceleGlove, the authors placed one more sensor on the back of the hand to improve the recognition process, and used 6 sensors mounted on gloves for recognition of 23 letters in the VSL alphabet and two postures for "space" and "punctuation". Users may spell each word and construct entire phrases using letters. This is very problematic when applied to real-world applications where translation speed is critical. After the testing, the recognition rate is high even when the postures are not executed correctly, such as when the finger is not entirely bent or the palm is not straight (Bui T.D, 2007) [4]. In other words, this approach can be effectively conducted in real-time regardless of external factors, which solve one of the problems we currently face is that the detected objects (e.g., the hands) have to be distinguished from the background, the higher contrast among the objects to the background as well as other objects the more confidence and accuracy the model gets, otherwise the highest accuracy will not be achieved and the model will have difficulty capturing the object. However, as we mentioned above, the biggest limitation of this method is that it is not practical when the users need to bring these gloves around all the time and wear it whenever they want to communicate with others.

2.2 Vision-based Approach

The vision-based approach is the method that scientists are currently researching and aiming for. These systems use cameras and software to capture and analyze visual information in the form of static or sequence photos without the third devices like gloves or sensors. The processed data then will be feed into a state-of-the-art Machine Learning or Deep Learning model to create a recognition system. It can be noticed that, the vision-based technique is more suitable for solving practical problems than the sensor-based approach, and more accessible for users. However, this domain faces numerous challenges under real-world condition, including complex and various backgrounds, fluctuations in lighting conditions, changes in skin color, camera qualities, occlusion, and so forth.

2.2.1 Support Vector Machine (SVM)

In 2017, a method for recognizing Vietnamese alphabet letters using static gestures and dynamic gestures for characters and diacritics respectively was proposed by a group of researchers from Vietnam and Canada [6]. Microsoft Kinect camera, a product manufactured by Microsoft is used to extract the movement of hands and head of the observed person (Mutto, et al., 2012) [7]. The result when inputting a dynamic gesture is represented by a sequence of RGB-D images (image that contains both color and depth information). Then, multi-class SVM is used to classify both static and dynamic gestures. After testing 30 predefined gestures with a total of 3000 patterns recorded, this system executed the accuracy from 80% to 95% [6]. However, this approach is hard to integrate into real world applications because of the constraints of the Microsoft Kinect camera's recorded distance is about 2.5-meter in front of the KinectV2, as well as the equipment requirements.

2.2.2 Deep Learning Techniques

A study in 2019 [8] proposed and compared the performance of two methods for VSL recognition directly from videos: the first one used the traditional classification SVM model with local descriptors (spatial, scene features); while the second one based on the combination of multiple Deep Learning techniques, including 1 layer of CNN with pre-trained VGG16 model and Long Short-Term Memory (LSTM) model, which they called the Deep Vietnamese sign language detection (DVSL). This experiment consists of two datasets totaling 27 words. The first has 12 words and presents the vocabulary of the relative family topic, which includes signs with minimal changes between frames or even static. Meanwhile, on the other hand, the second one consists of 15 words in the same topic and has a more complex and longer sequence of actions which have the relative direction of the hands and body parts. In other words, the motions of the second dataset will be more difficult to detect and classify compared to the first one. A video which describes the gesture of each word will be used as the input to the model, each frame will be extracted manually, a skin detection technique is used to extract and focus on the movement of hands in each frame before applying the SVM/DVSL method. The workflow of the DVSL is presented in Figure 2.1.

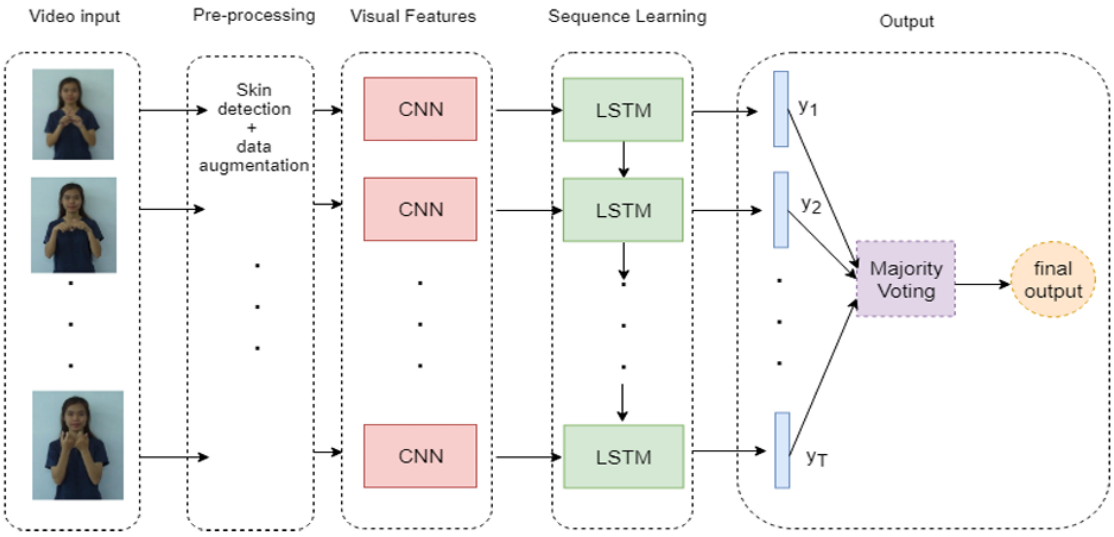


Figure 2.1: Workflow of DVSL System. [8]

The research indicates that the DVSL model achieved better performance on the first dataset compared with the SVM model. In detail, 95.83% is the highest accuracy achieved from the DVSL model, 10% higher than the accuracy of the SVM classifier. However, take the second dataset as consideration, DVSL model is not as good as the SVM models (86.67% compared to 90.16%) because the capability of feature extraction of the CNN model is limited when capturing gestures with complex motion which relates to the relationship between body parts (Anh, et al., 2019) [8]. Moreover, the authors point out that the ability to identify the differences from the continuous frame in sequence images exported from a video also affect the output accuracy of the model.

Two years later, in 2021, a sign recognition system was proposed by Van.Q.P. and Thanh.B.N. [9] that used the same dataset from the research in 2019 [8]. Firstly, the pre-processing stage, which is intended to solve the limitations of the DVSL method (handle the complicated gestures and filter the changes from each frame in sequence images). Finding movement zones and extracting movements are the two steps of the "Bounding box observed object using Gauss distribution" approach that Dr. Van and his team create. By automatically selecting specific frames where the motion of the objects differs dramatically from the preceding frame, the authors were able to standardize input films. The amount of data that the network must handle is reduced thanks to this effort. Additionally, the authors note that it is inefficient to use the full video input's default size [9]. In order to remove comparable motions from the input video, the correlation coefficient has been utilized to determine whether motion in continuous frames is similar. The Figure 2.2 shows the result before and after selecting and focusing on the movement.

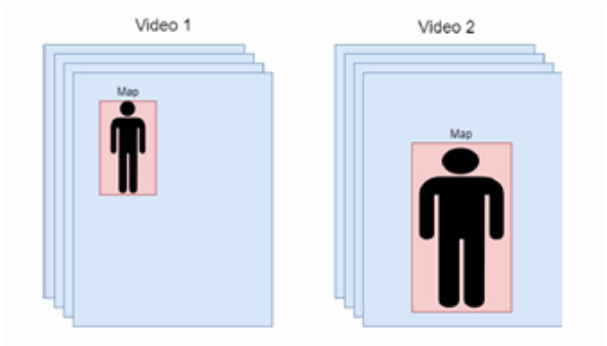


Figure 2.2: Gauss distribution bounding box observation. [9]

In order to identify the class to which each video belongs, they then build a model in the deep learning and classification stage that combines GoogLeNet and BiLSTM. GoogLeNet, a CNN pre-trained model, extracts features from each frame of sequence movies, whereas BiLSTM classifies signs. The results of training with a dataset of 2700 samples representing 27 VSL signs indicating daily activities and words such as mother, father, cloud, sun, eat, drink, and so on. With an accuracy of up to 99.38% with the validation set and 98.15% with the test set, the experimental results increased by 10% when compared to the 2019 study [8] that combined the VGG16 and LSTM model. The comparison between the proposed method from Van.Q.P. and Thanh.B.N. with the other methods is illustrated in Figure 2.3.

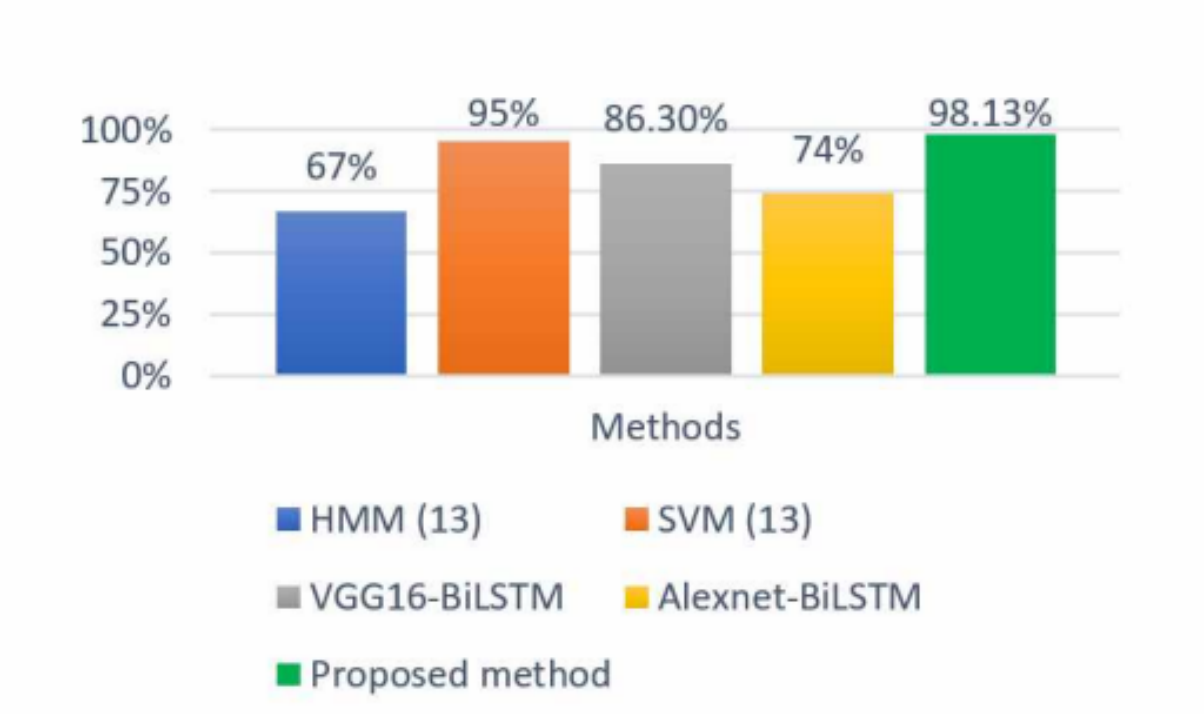


Figure 2.3: Accuracy comparison of the proposed method in 2021. [9]

The only drawback to this concept is the processing time. According to the researchers, the average time for video recognition—which is 4-5 seconds for pre-processing and 1-2 seconds for video classification—exceeds the standard for a real-time application when utilizing a complicated model like the BiLSTM [9]. As a result, this model still does not meet the necessary conditions to be implemented and used in the real world, where the detection step must be estimated in milliseconds.

Overall, these studies contribute to the advancement of sign language recognition systems by combining deep learning algorithms based on CNN with Long Short-Term Memory (LSTM) or Bidirectional LSTM (BiLSTM) model to improve communication accessibility for people with speech or hearing disabilities. In this study, we will optimize and emphasize the advantages of the YOLO11 algorithm, the state-of-the-art in the Ultralytics YOLO series, proposed real-time VSL detection works by recognizing the users' hand motions from a video without the usage of an additional equipment. Moreover, the mentioned drawbacks of previous works, which are related to handling complex gestures and processing time, could be overcome. In the meantime, our research tries to advance this research by exploring the application of live video or hand gestures via the camera.

2.2.3 Fusion of Heatmap and Depth information

In 2024, there were two papers focus on different methodologies for Vietnamese sign language and hand gesture recognition, employing vision-based and a combination of sensor-based and vision-based approaches respectively.

The first study proposes Ad2C, a novel framework for isolated Vietnamese Sign Language (VSL) recognition that fuses heatmap and depth information using 3D Convolutional Neural Networks (3D-CNNs) [10]. The authors highlight that sign language is the primary means of communication for deaf and hard-of-hearing individuals, involving the upper body through hand gestures, facial expressions, and body language. Research in this field is motivated by the potential to empower this community, facilitate seamless communication

with technology, and promote inclusivity. Sign language recognition poses challenges within computer vision. Existing studies explore various modalities like RGB frames, optical flows, audio waves, and human skeletons for feature representation. Skeleton-based action recognition has gained attention due to its focus on action and compactness, being immune to contextual noise like background or lighting changes. Graph Convolutional Networks (GCN) are popular for skeleton-based recognition, representing sequences as spatiotemporal graphs. However, GCN methods have limitations, including sensitivity to small coordinate perturbations and difficulty combining with varying modalities like RGB and depth. Convolutional Neural Networks (CNN) are also used for skeleton-based recognition, sometimes modeling sequences as pseudo-images or stacking heatmaps, but these methods can suffer from information loss.

To address these difficulties, the paper proposes the Ad2C framework. Ad2C accepts 2D poses represented by stacks of heatmaps of skeleton joints as input, rather than operating on coordinates. A 3D heatmap volume is created by stacking heatmaps over time and depth data also be used in the Ad2C. The Ad2C architecture is an improvement and combination of two CNN networks based on the I3D structure (see Figure 2.4 for more details). It uses two flow 3D-CNNs for processing the 3D heatmap volume and depth data. The model is improved by splitting the Heatmap and Depth data into two separate training streams before combining their features. This structure is described as Mid-Fusion in the experiments, as demonstrate in Figure 2.5. The authors also tested a variant combining data earlier without separate streams, but this was less feasible. Ad2C's key distinctions from GCN-based techniques include its input format (Heatmap Volumes and Depth Volumes vs. Skeleton Coordinates) and architecture (3D-CNN vs. GCN), making it less sensitive to pose estimates and easier to combine modalities.

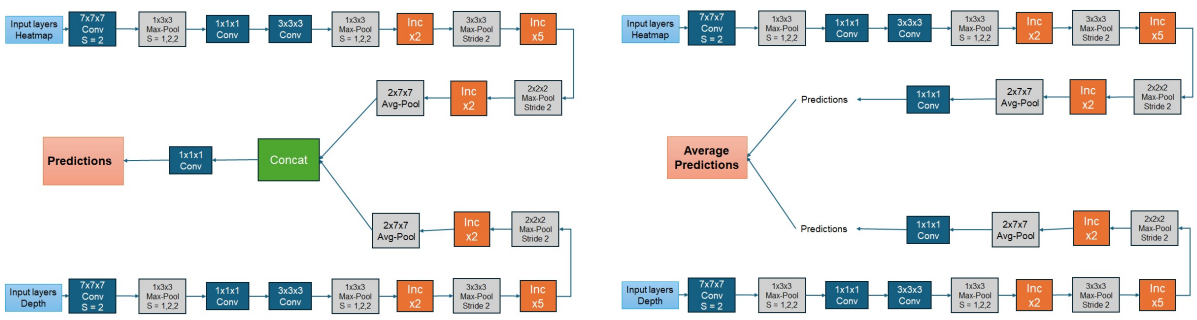


Figure 2.4: Ad2c Framework. [10]

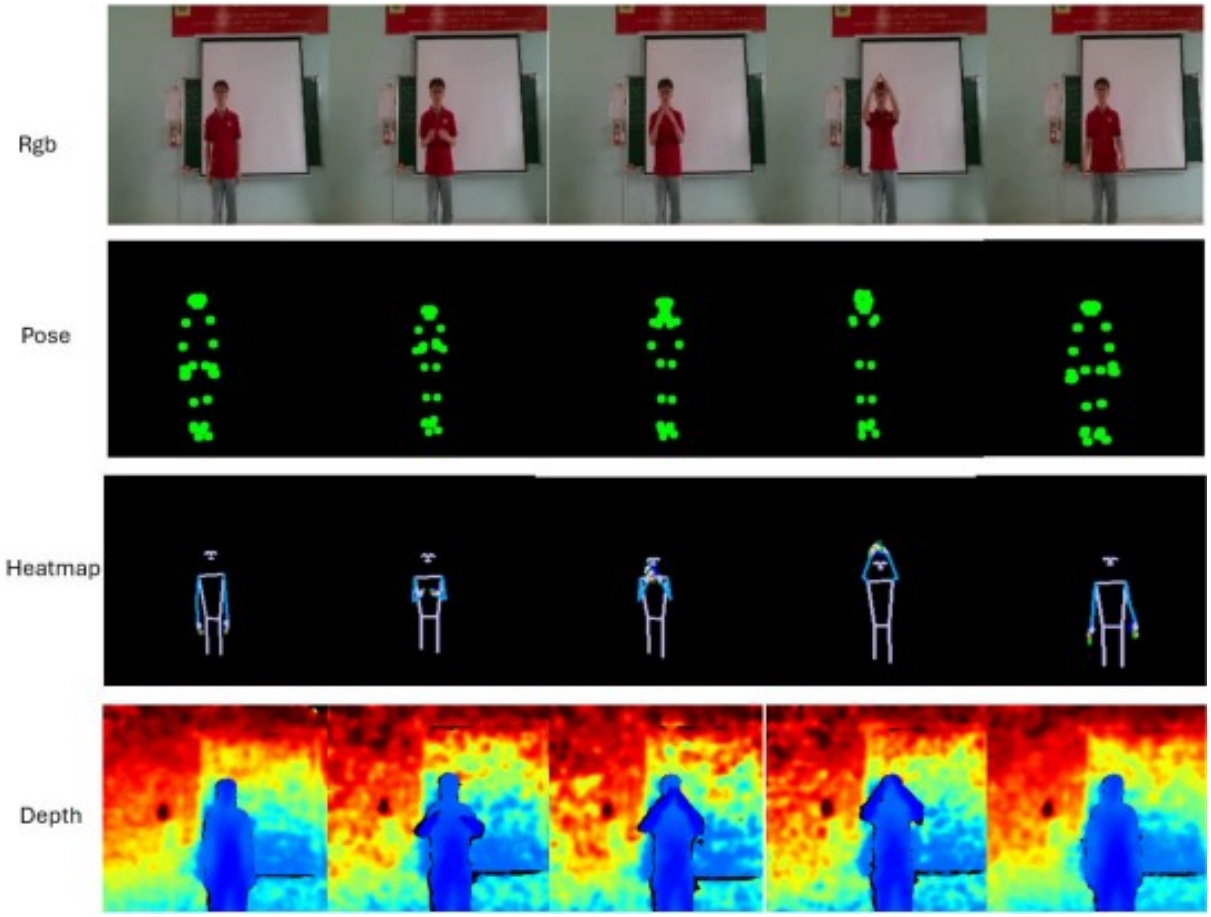


Figure 2.5: Pose estimators produce 2D poses that are represented by stacks of heatmaps of skeleton joints. [10]

For training, the objective is a multi-task optimization problem with components for localization, class confidence, and activity confidence loss. The authors collected their own dataset using RGB-D cameras. The dataset consists of 120 isolated Vietnamese words performed by 110 volunteers who learned the gestures from instructional videos. The dataset includes 18948 videos after preprocessing.

Advantages of Ad2C include robustness to small coordinate perturbations, ease of combining with other modalities, superior speed and accuracy compared to prior methods, and reduced sensitivity to environmental factors like lighting due to heatmap inputs. However, inference time increases significantly with larger input sizes, and some CNN-based skeleton recognition methods suffer from information loss during processing. The model achieved 80.15% accuracy using combined heatmap-depth inputs, outperforming I3D models on RGB-D, RGB, and heatmap inputs alone (see Table 2.1 for more detail). This approach can eliminate the environmental factors like lighting and background variations which is superior compared to YOLO11 and other deep learning methods. Compared with other vision-based approaches, this model output a much lower accuracy, however provide a faster inference time.

Table 2.1: Ad2c performance compare with the other methods [10]

Model	Input	Accuracy	mAP	Validation Loss
I3D	RGB	59.35%	0.581	1.65
I3D	Heatmap	53.33%	0.548	1.85
I3D	RGB-D	64.21%	0.633	1.52
Ad2C	RGB-D	79.80%	0.804	0.77
Ad2C	**Heatmap-Depth**	**80.15%**	**0.825**	**0.74**

2.3 Combination of Sensor-based and Vision-based Approach

Another paper in the same year of 2024, presents a sensor-based combine with vision-based hand gesture recognition system using bio-impedance measurements collected via electrodes placed on the back of the hand and fingers. The researchers note that body language, including hand gestures, is very important for communication between people and between people and machines. Automated systems in industries like manufacturing and healthcare could benefit from better human-machine communication, which includes recognizing gestures for tasks like controlling robots or assisting in physical therapy [11].

The device developed in this study uses a Raspberry Pi computer for control and processing. The device analyzes the voltage that results from sending a little electrical current through electrodes into the hand in order to determine resistance. The device measures signals from various hand points consecutively with the use of a component known as a multiplexer. This device is utilized by using the dry conductive fabric, which is lightweight, resilient, and reusable, to make sure that the electrodes can make contact with the skin. Five electrodes, including the thumb, index, middle, ring, and pinky — are positioned on the fingers, and one in the center serves as a reference point. Two electrodes are used to provide the current, while six electrodes are used to receive signals. For flexibility, the electrodes on the fingers are positioned at the second joint. For bio-impedance measurements in hand gesture recognition applications, the electrical impulses are supplied at a predetermined frequency of 50 kHz. The AD5933 chip uses a voltage-controlled current source to regulate the current amplitude and guarantee safe operation for human applications. Figure 2.6 illustrates a simplified functional block diagram of the system.

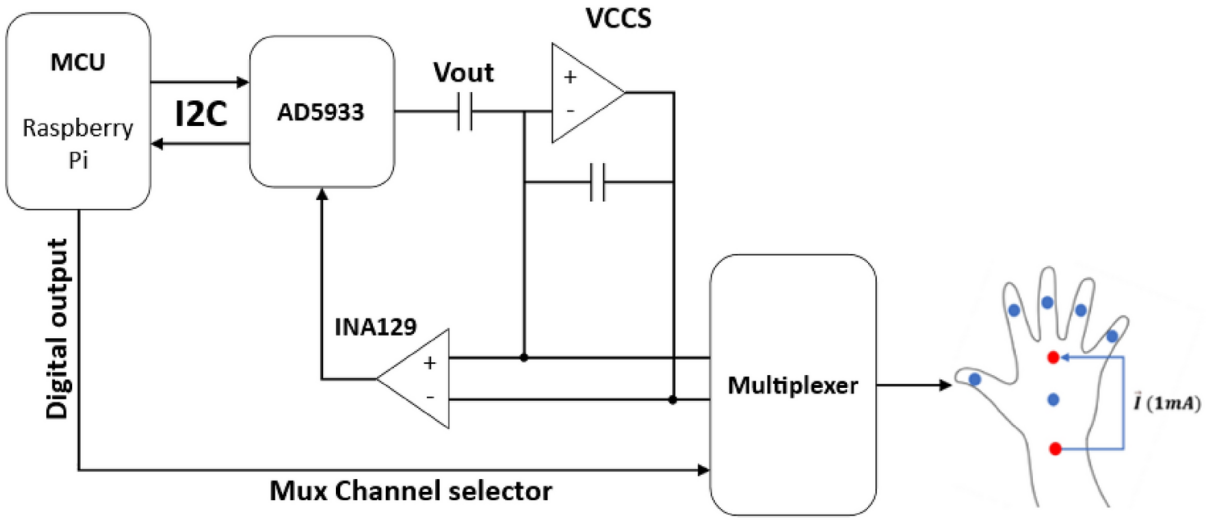


Figure 2.6: Bio-impedance-based gesture recognition system. [11]

To recognize gestures, the system uses machine learning models including CNN, XGBoost, and Random Forest were trained on the impedance data. The experiments involved ten deaf individuals performing nine different hand gestures. Among these, 6 labels "Agree", "Eat", "Drink", "Sleep", "Sorry", and "Thanks" are dynamic, while 3 labels "A", "B", and "C" are static. Data was collected for about 1.5 seconds per gesture, resulting in 95 impedance values per finger. The trained model is then made smaller using TensorFlow Lite so it can run efficiently on the less powerful Raspberry Pi for real-time application [11]. According to the researchers, compared to camera-based systems, the bio-impedance technique has the following benefits: portability, cost-effectiveness, robustness to environmental conditions like lighting, improved privacy, and lower power and computing requirements. The study was then tested on three different training strategies to compared the results and gave out the current limitations. The subject-dependent training strategy, the model is trained and tested on data from the same subject, yielded high accuracies when using Random Forest and CNN, at 97.24% and 95.12% consecutively, as seen in Figure 2.7.

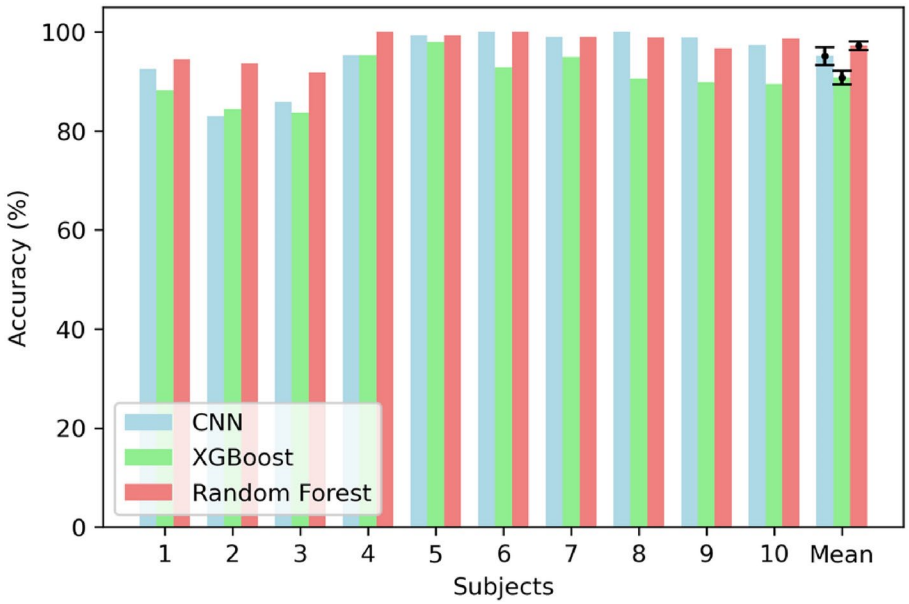


Figure 2.7: Classification accuracy of CNN, XGBoost, and Random Forest models using subject-dependent strategy. [11]

During real-time testing, the system using the CNN model took about 400 ms to recognize a gesture. The real-time accuracy was lower than the offline tests, at 77.8%, likely because the model encountered data it hadn't seen during training. Training the model on a wider range of data, including different environmental conditions, could improve real-time accuracy.

On the other hand, the subject-independent strategy requires the model to be trained on data from a group of subjects and tested on unseen subjects, performed poorly, with average accuracies around 30-35% for all three models (Figure 2.8). This indicates that it is difficult for bio-impedance models to generalize well across different individuals due to variations in their bio-impedance signals.

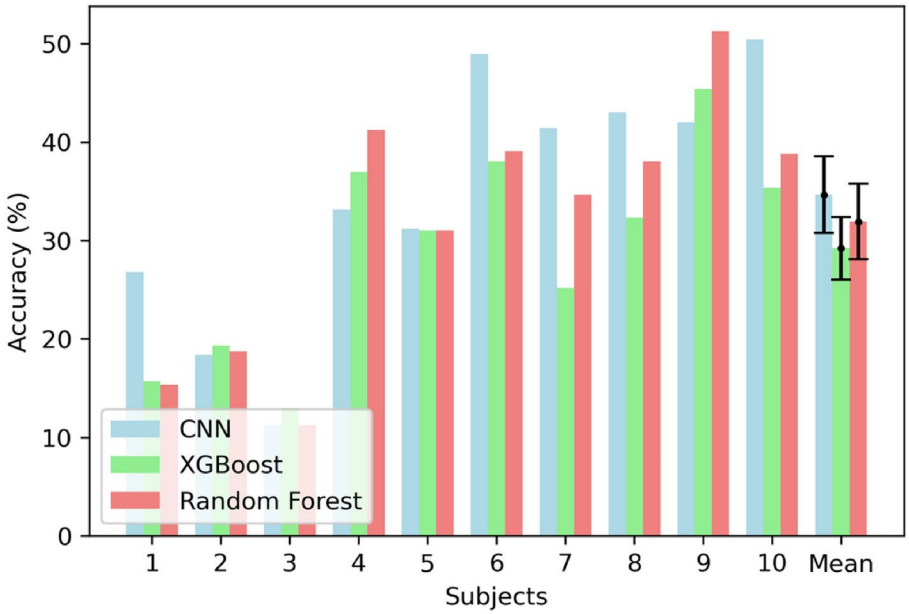


Figure 2.8: Classification accuracy of CNN, XGBoost, and Random Forest models using subject-independent strategy. [11]

And finally the mixed-subject strategy, the model is trained on multiple subjects, partially fine-tune on the new subject and it requires some data from the new user for partial training or fine-tuning before using it. This strategy showed better generalization than the subject-independent approach, with the CNN model performing best with an average accuracy of 87.1% using five-fold cross-validation (Figure 2.9).

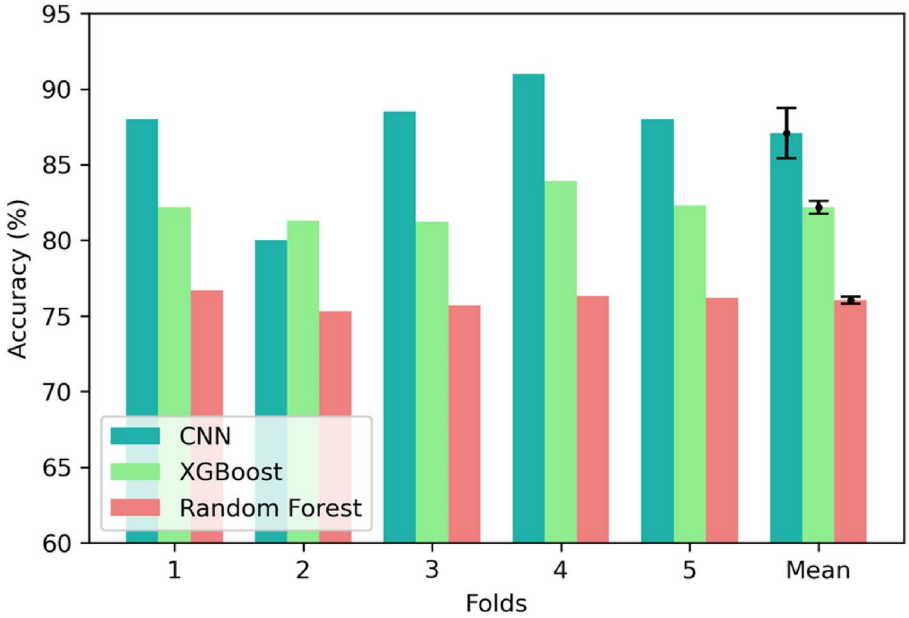


Figure 2.9: Five-fold cross-validation accuracy of CNN, XGBoost, and Random Forest models using the mixed-subject strategy. [11]

In conclusion, the subject-independent and mixed-subject techniques performed noticeably worse than the subject-dependent strategy, indicating challenges in generalization. The researchers also acknowledge several factors have a negative impact on the results with the current device. The limitations including the variation in skin impedance during

gestures is minimal, which makes recognition challenging. The calibration process is complicated and requires users to remove electrodes, which is inconvenient. The current electrodes need adhesive tape for stability. Also, factors like sweat and temperature can affect skin conductivity and thus impact the accuracy [11].

Researched on the recent works on Vietnamese Sign Language (VSL) recognition over the last decade have explored various deep learning techniques to improve feature extraction and robustness recognition system to deal with the variation of environmental factors, as well as the other limitations. These vision-based approaches, which use 3D algorithms, CNNs, and multi-stream networks to capture the patterns of gestures from videos and images, show better accuracy and real-time performance on VSL datasets [9][10]. In comparison to deep learning techniques, traditional methods that use SVM also show positive results, although they are typically less adaptable when managing complex motion dynamics [8]. Our recent work, which study on the YOLO11 model for hand gesture recognition in Vietnamese Sign Language, expands on existing vision-based methods by utilizing an advanced object detection framework built for high precision and real-time performance. Like other vision-based techniques, YOLO11 might still have issues with lighting, occlusion, and background variations, which will need to be addressed with careful dataset preparation and augmentation.

In conclusion, by utilizing state-of-the-art deep learning object identification techniques, the YOLO11 model and our suggested YOLO model promise to provide a more scalable, effective, and user-friendly solution for Vietnamese Sign Language recognition as compared to earlier sensor-based and vision-based approaches. It represents a significant breakthrough in the field of sign language recognition systems by achieving a balance between the trade-offs of accuracy, cost, and usability.

CHAPTER 3

METHODOLOGY

This section describes detailed the methods used to achieve the defined objectives of this project. From preparing the VSL dataset and applying data augmentation during pre-processing stage. Then, proposed a new YOLO structure based on the YOLO11 model during the deep learning and classification stage. To train our model, first we conduct a training data preparation phase. Then we train the model. Once trained, in the inference phase, our model will get a video as input, feeding the video into the trained model to recognize the words corresponding to the signs in video frame. Finally we conduct a postprocessing to generate sentence from the recognized words. The system will run through a Transformers Model for Vietnamese Paraphrasing to create a completed Vietnamese sentence from given words. The trained YOLO model can also be used to detect hand motions and recognize gestures in real time through a camera or webcam. In the following subsections, we will discuss more details about each part in our system.

3.1 Data preparation

Figure 3.1 illustrates the training dataset preparation pipeline that transforms raw image collections into ready-to-use training data. The system begins with image upload and labeling, then applies preprocessing steps. To enhance the dataset and improve model robustness, it performs data augmentation through various techniques, before generating and exporting the final augmented dataset for the model training.

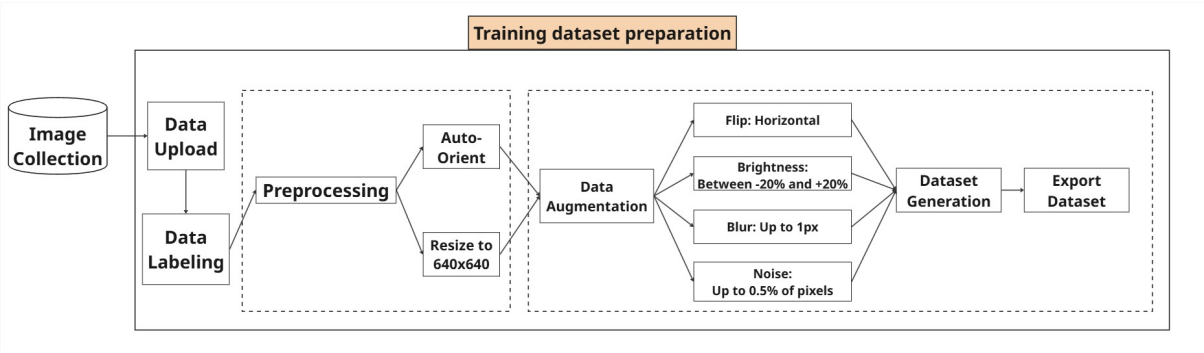


Figure 3.1: Training dataset preparation.

3.1.1 Data Collection

Vietnamese Sign Language (VSL) is based on the established American Sign Language (ASL), but VSL also has some special signs which are not included in the ASL dictionary, such as the new character "crossed D" and six Vietnamese diacritic marks: "Grave", "Acute", "Hook", "Tilde", "Under-dot", and "Breve" (Hai P.T, et al., 2018) [12]. Additionally, there are instances in which a single term in the English vocabulary system represents a series of distinct Vietnamese words. A study in 2023 by Carl Börstell at the University of Bergen indicates that unrelated sign languages often share a baseline lexical similarity of approximately 20–30%, primarily due to shared iconic motivations [13].

Despite the growing interest in sign language recognition systems, there is currently no publicly available dataset for Vietnamese Sign Language (VSL) suitable for training

models. This significant gap in resources poses a challenge for developing accurate and robust detection systems tailored to the Vietnamese linguistic and cultural context. To address this limitation, the creation of a bespoke dataset is necessary. Consequently, this study involves self-recording and photographing gestures performed by the researcher, followed by meticulous labeling to construct a comprehensive VSL dataset. This approach ensures the dataset's relevance and specificity to the project's objectives, enabling effective model training and evaluation.

After filtering out some gestures using the Vietnamese Sign Language Catalog provided by QIPEDC [14], we created a dataset consisting of 20 words which represents some common words that people use on a daily basis, including: "Bạn", "Tôi", "Đói bụng", "Khát nước", "Tên", "Xin chào", "Tạm biệt", "Yêu", "Có", "Không", "Cảm ơn", "Xin lỗi", "Ăn", "Uống", "Là gì?", "Bố", "Mẹ", "Giúp", "Không sao", "Việt Nam". Each word will have a different level of complexity, some only have small changes between frames, some have the relative position and orientation of motion gestures and the body parts.

3.1.2 Data Annotation

After taking about 100 photos of each sign, the collection then will be uploaded to Roboflow [15], an end-to-end computer vision platform that simplifies the process of building computer vision models, to label data for a YOLO11 object detection, instance segmentation, or classification model. While it can be used for classification, segmentation, or real-time detection tasks, our project specifically utilized Roboflow for annotating a custom dataset of VSL gestures. For reasons of ensuring the accuracy of the dataset as well as to check if any images are blurred or smudged, all the images will be labeling manually by hand using the Annotation Editor provided by Roboflow [16]. Figure 3.2 shows an example of using the Annotation Editor to label a sample.

Figure 3.2: Roboflow Annotation Editor. [16]

3.1.3 Data Augmentation and Preprocessing

We used a number of data augmentation techniques to artificially increase the dataset size while adding more variability because the original dataset included 1,638 high-quality images. Our dataset has grown by approximately three times as a result of this

augmentation process, and the model's capacity to identify hand signs in various scenarios has been greatly enhanced. These are some augmentations we used to expanded our dataset:

- Horizontal Flip: In order to replicate real-world situations where the camera might appear in various orientations, we used horizontal flipping. This improved the model's ability to generalize to hand signals that show up on the right or left side of an image.
- Brightness: We increase or decrease the brightness in each image randomly between -20% and +20% in order to improving the model's ability to recognize hand signs under different light conditions.
- Blur: Since this project's objective is to handle the real-time detection, the factor like camera should be take into consideration where can cause the objects blurring during live detection. For example, the camera is stationary, but objects it is detecting are often moving; the camera is moving, but objects it is detecting are stationary [17]. A camera and its objects its detecting are moving. Therefore, we applied random blurred up to 1 pixel.
- Noise: We add salt-and-pepper noise up to 0.5% of pixels to help the model be more resilient to camera artifacts.

Preprocessing Techniques:

- Auto-Orient: Applied - To ensure uniformity in all images and consistency of orientation to avoid situations like some images may have stored incorrect and incomplete dataset which could confuse the model.
- Resize: Stretch to 640×640 - resized images to the same size, in case that not all images will be in the exact same resolution which helped model differentiate hand sign even in different size and formats.

3.2 YOLO11

3.2.1 YOLO Overview

In computer vision, YOLO (You Only Look Once) is a well-known and effective object recognition algorithm. It is intended for real-time object identification and localization within a picture or video frame. It is a single-stage object detector that predicts the bounding boxes and class probabilities of objects in input photos using a convolutional neural network (CNN) [18]. YOLO is incredibly quick and effective because it uses a single neural network to process the entire image, unlike typical object detection techniques that frequently require several passes over the image [19]. The real-time object detection system YOLOv1 was first released in 2015 and quickly expanded through several iterations, each improving on the one before it to address limitations and enhance performance. YOLO11 was released on September 30^{th}, 2024 by Glenn Jocher (Ultralytics) [2]. See Figure 3.3 to see the history of the evolution YOLO algorithms.

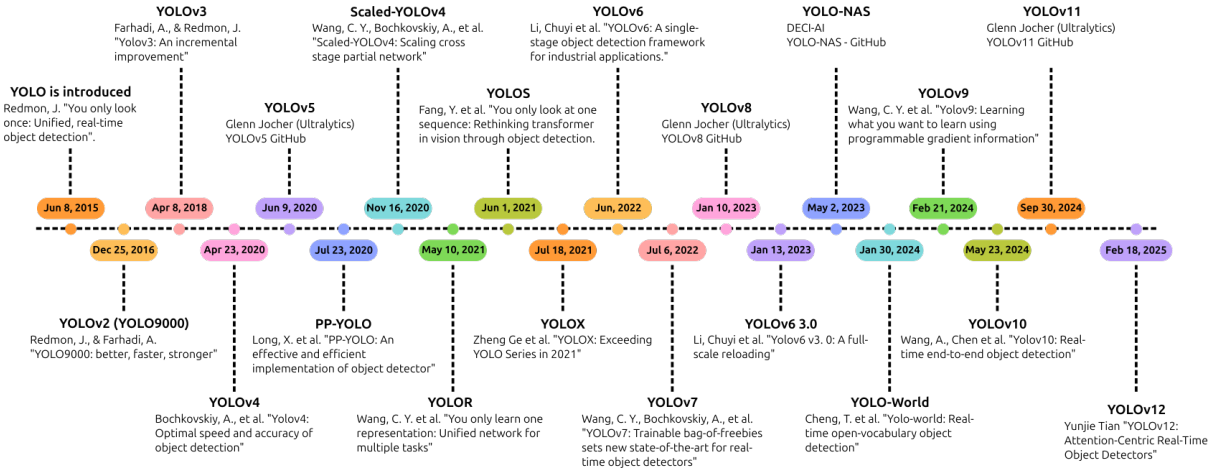


Figure 3.3: Evolution of YOLO throughout the years. [18]

Segmenting an image into a grid of cells, rather than smaller images, is the basis of the YOLO algorithm. The picture is divided into a S × S square grid. Every Grid cell forecasts the confidence score and B bounding boxes. The model's confidence that an object is in the predicted box is represented by the confidence score, which is determined as follows:

$$Confidence(C) = P(Object) \times IoU_{truth,pred}$$

P(Object), which goes from 0 to 1, is the object's presence probability. In this case, 0 denotes the absence of the object and 1 denotes its probable presence. The intersection over union measure or $IoU_{truth,pred}$, is the ratio of the intersection area to the union area of the predicted bounding box and the ground truth bounding box. Figure 3.4 shows the examples of the IoUs based on their values.

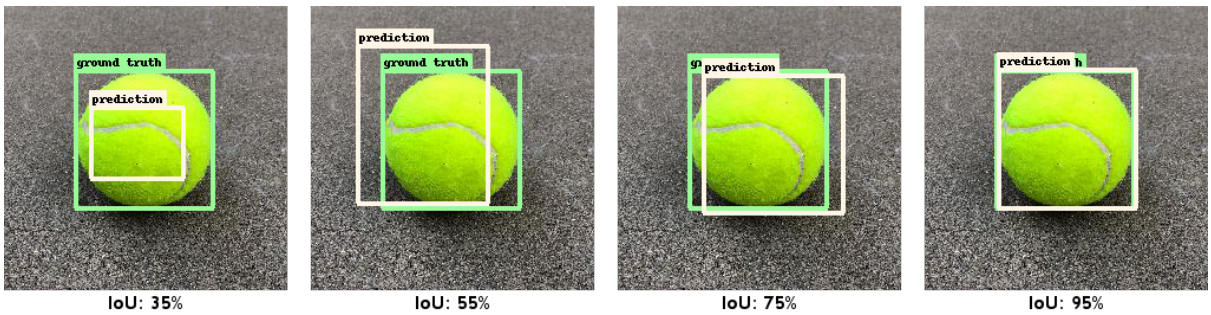


Figure 3.4: IoU Examples. [20]

A bounding box is defined by the following five elements: x, y, w, h, and confidence score. The center of the box with respect to the grid cell's edge is represented by the (x, y) coordinates. The bounding box's width and height are denoted by the letters w and h. Finally, the IoU between the projected box and any ground truth box is represented by the confidence prediction.

Since the release of YOLOv8 in 2023, there are several relevant academic studies were identified that apply YOLOv8–YOLOv11 (or custom model based on these YOLO's algorithm) to hand sign language detection on different languages (ASL, Bangla/Bengali, etc.). The results of the previous research on sign language detection using YOLO models were very impressive, showing the potential of using YOLO model for this study's objective. Table 3.1 summarizes each study's methods, datasets, and key results.

Table 3.1: Studies Using YOLOv8–YOLOv11 for Hand Sign Language Detection (2023–2025)

YOLO Model	**Dataset**	**Performance**
YOLOv8 + MediaPipe hand landmarks [21]	Custom ASL (Lexset + Kaggle + Roboflow, 29,820 images, 26 classes)	Prec: 98%, Rec: 98%, F1: 99%, mAP@0.5: 98%, mAP@[.5:.95]: 93%
SLR-YOLO (YOLOv8 + RFB + BiFPN + Ghost + Cutout) [22]	ASL + Bengali Alphabet (public datasets)	Acc: 90.6% (ASL), 98.5% (Bengali)
YOLOv8 vs. YOLOv5 [23]	Custom ASL Phrases (22 classes, 2.2K images)	YOLOv8 Prec: 95.6%, YOLOv5: 92.8%
YOLO11 + MediaPipe [24]	ASL alphabet (130K frames)	Prec: 98.5%, Rec: 98.1%, F1: 99.1%, mAP@0.5: 98.2%
YOLO11 [25]	Bangla + English alphabet (64 classes, 9,556 images)	Prec: 99.12%, Rec: 99.63%, F1: 99.37%, mAP@0.5: 99.4%
YOLOv9e / YOLOv9c variants [26]	ASL alphabet (26 classes, 1,292 images)	YOLOv9e mAP@0.5: 97.84%, YOLOv9c: 96.46%
YOLOv10 (custom-trained) [27]	Bangla static signs (14 classes, 1,949 images)	Prec: 100%, Rec: 98%, F1: 86%, Acc: 90.67%

The results of different versions of YOLO are compared to find the best-performing model among the same. In this section, we have compared the performance metrics as mAP@0.50, Precision, F1-Score, Recall and Loss function to propose the best-performing model. Table 3.2 presents the results of training YOLO11, YOLOv10, YOLOv9, and YOLOv8 for 100 epochs on our self-created VSL dataset. We utilized the Ultralytics library for this training, and the results demonstrate that YOLO11, being the latest version in the YOLO series, has achieved the most impressive performance.

Table 3.2: Comparison of YOLO Models Metrics

Metrics	**YOLO11**	**YOLOv10**	**YOLOv9**	**YOLOv8**
GPU Memory Usage (GB)	8.080	9.160	9.030	7.190
Bounding Box Loss	0.475	0.915	0.435	0.488
Classification Loss	0.267	0.456	0.245	0.262
Distance-Focal Loss	1.004	1.919	0.979	1.043
Total Loss	1.728	3.290	1.659	1.793
Precision	0.995	0.966	0.962	0.987
Recall	0.984	0.907	0.936	0.988
mAP50	0.995	0.987	0.988	0.993
mAP50-95	0.817	0.806	0.814	0.812
F1	0.989	0.935	0.948	0.987

Notably, YOLO11 incorporates several advancements in architecture and optimization techniques, which contribute to its enhanced accuracy and efficiency compared to its

predecessors. Because of its outstanding performance on our VSL dataset compare to the other recent YOLO models, YOLO11 was used for further investigation in this study.

3.2.2 YOLO11 Architecture

YOLO11's structure is constructed by three main parts: the Backbone, the Neck and the Head. The Backbone is a CNN deep learning architecture which basically acts as a feature extractor; the Neck combines the features acquired from the various layers from the Backbone model. It typically involves up-sampling and concatenation of feature maps from different levels; the Head predicts the classes and the bounding box regions which is the final output of the object detection model. Figure 3.5 shows the workflow of our implementation from the beginning to the end with a brief overview of the YOLO object detection algorithm.

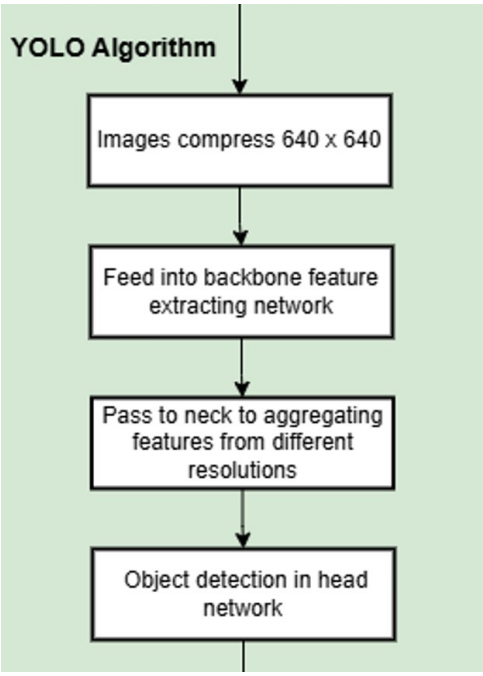


Figure 3.5: YOLO Algorithm.

The detailed information about each phase in the YOLO algorithm has been provided in Figure 3.6 by Dr. Priyanto Hidayatullah and Dr. Refdinal Tubagus [28] based on the YOLO11 architecture file which is located in the Ultralytics' GitHub [29]. With certain additional integrations and parameter adjustments, the YOLO11's design is essentially an upgrade over the YOLOv8 [28]. Particularly the application of the recently developed C3k2 (Cross Stage Partial with Kernel Size 2), SPPF (Spatial Pyramid Pooling - Fast), and C2PSA (Convolutional block with Parallel Spatial Attention) blocks, which enhance feature extraction and other aspects of the model's performance [30].

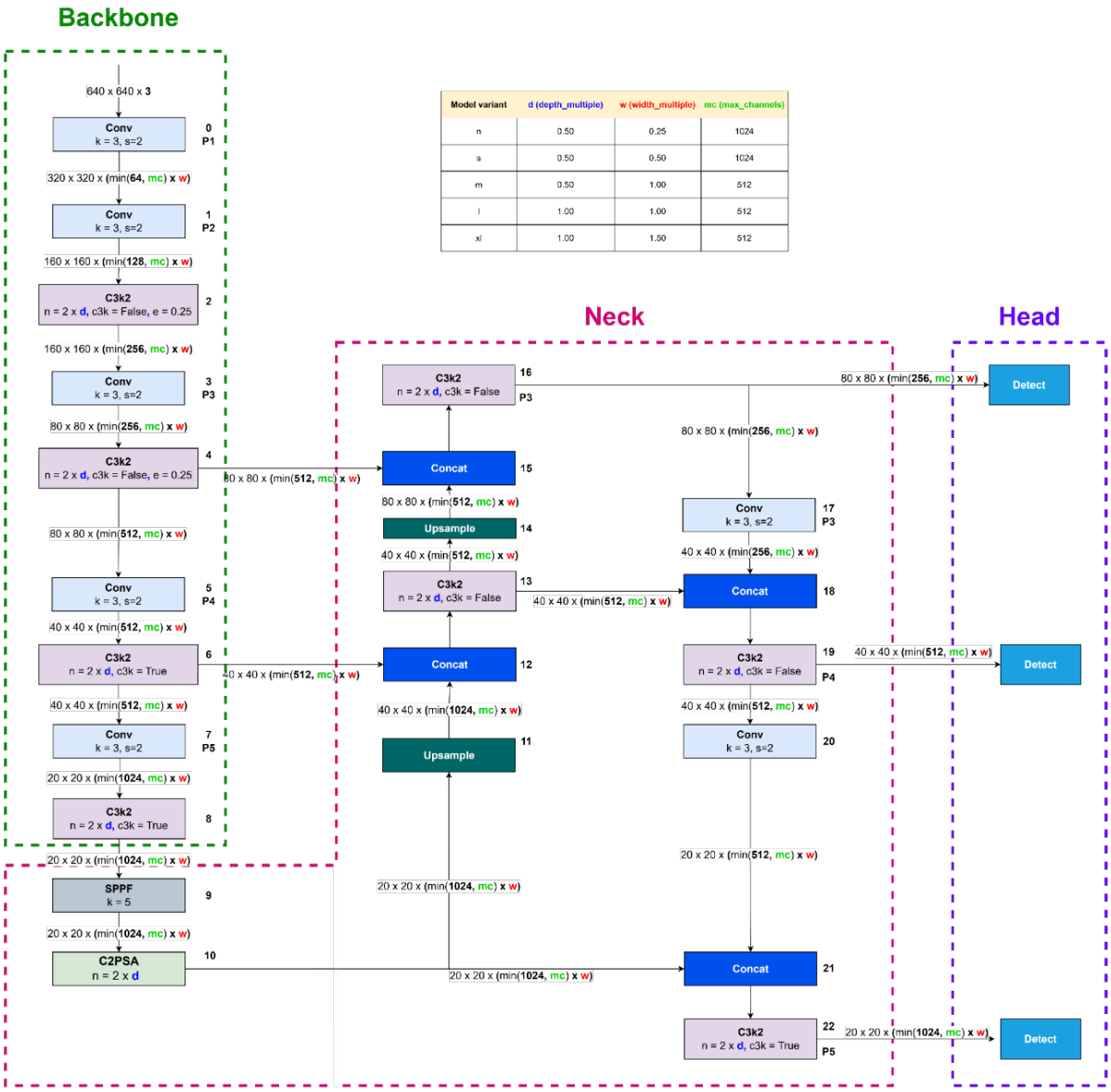


Figure 3.6: YOLO11 architecture with the new C3k2, SPPF blocks and the C2PSA module. [28]

The YOLO11 has five available architectural versions scaled from nano to extra large [2], the application of each model depends on the size of the project. The main difference of each variant is defined by the three parameters which determine the number of blocks in each part. Each following model is larger in size, trained on more data, and takes longer to compute. This means that smaller models will be more compact and faster to use, but accuracy may suffer. For this research's use case, a small YOLO11 model (YOLO11s) will be utilized.

- **Backbone:** First the input of the YOLO11 to the Backbone is an image in size 640 x 640 pixels with three channels. The Backbone is made up of numerous convolution layers that extract distinct features at various resolution levels. The convolutional blocks will reduce the resolution of the image into half when going through. Next is the C3k2 block, which is newly introduced in YOLO11, this block is a custom implementation of the CSP Bottleneck and used to control the capacity of the Bottleneck block. CSP networks split the feature map and process one part through a bottleneck layer while merging

the other part with the output of the bottleneck. This reduces the computational load and improves feature representation [30].

- **Neck:** The images then will be passed to the Neck, YOLO11 focuses more on spatial attention through C2PSA, which helps the model to focus on the key regions in the image for better detection, this also a new block in this architecture [30]. The Upsample layer is used to increase the map resolution in YOLO11, using the nearest neighbor method. This method works by repeating the values of nearby pixels or elements of the image or input tensor to fill in the newly generated pixels in a larger image. By this way, this layer will increase the feature map resolution of the C2PSA to match the feature map resolution of the C3K2 blocks. In addition to this, the Upsample layer will be combined with the feature from the C3K2 block in the Backbone using Concat.
- **Head:** The combination will go through another C3K2 block and the output will be the input for the detect block in the Head, this block is specialized for detected objects, relative to the image or video. Tailored for detecting VSL gestures across different scales (small, medium, large), these blocks lever- age refined features from the Neck module to out- put precise bounding box coordinates and class predictions.

3.3 Proposed Model

In this study, we propose a new YOLO architecture (See Figure 3.7), an improved object detection model based on YOLO11, with a focus on general object detection (hand sign) and reduced model complexity.

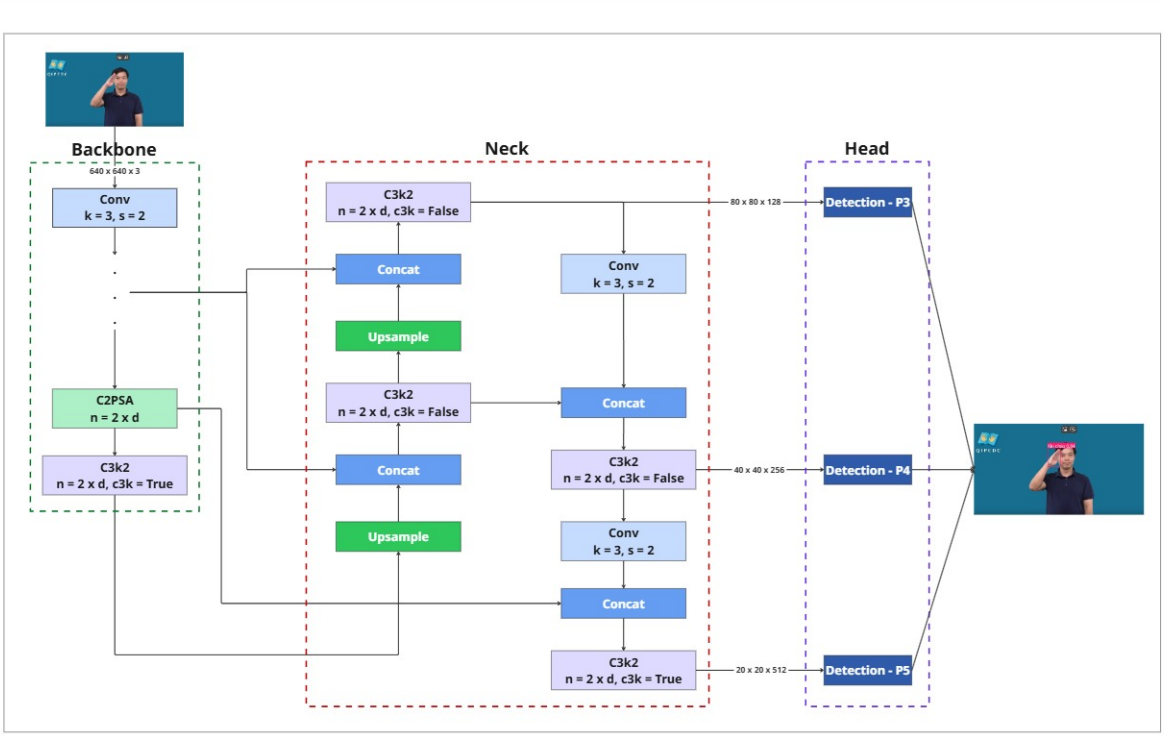


Figure 3.7: Proposed YOLO Model.

3.3.1 Motivation for Modification

Currently, the original model has 2 main limitations when dealing with hand sign detection:

- Downsamples too much → Hand's details may lose.
- Uses large feature maps (1024 channels) → Overkill for hand signs and cost resources.

Therefore, our proposed method aims to improve its network architecture and reduces the unnecessary complexity while still capturing enough hand features for higher identification accuracy. The specific flow of the original method compared to the proposed method are shown in Figure 3.8, the C3k2 block with the yellow highlighted is the new block we added to the Backbone architecture, as well as the model's scale reduced by half.

3.3.2 Architectural Modifications

To improve the model's performance, we reduced the number of channels from 1024 to 512. This represents a major improvement in how the model is built. We accomplished this by using smaller filters at the beginning of the network (32 filters instead of 64 filters). This change makes the model less complex while helping it work better. This modification helps the model balance its feature maps more effectively, which leads to better accuracy when detecting hands. The change also removes unnecessary parts of the network and prevents the model from keeping too much detailed information that might not be useful. By keeping only the most important features and making the detection process simpler, our model can work better with new data it has never seen before. The tweaks we met on YOLO11 architecture to handle our current application is as follow:

Changes to the Backbone

A key improvement in the head layer is the addition of a C3k2 block following the C2PSA attention module. The C2PSA module works by using attention mechanisms to give more importance to certain parts of the feature maps. It does this by identifying which information is most critical for the task. After the C2PSA module processes the features, the C3k2 block takes over to work with these adjusted features. This additional block helps combine and merge the important information that the attention module identified. The C3k2 block acts like a refining tool that smooths out and improves the signals that have been enhanced by the attention mechanism. This creates stronger and more reliable representations of the features, which helps the final detection part of the network perform better.

Improvement on Neck and Head layer

We also found that increasing the depth of the C3k2 block improves the model's ability to detect small hand features. Instead of using two layers, we changed it to use three layers before the final detection stage. This deeper structure allows the model to process features more thoroughly. With this deeper setup, the model can better handle detection at multiple scales, which increases overall accuracy. The extra layer ensures that the model can capture the fine details that are necessary for accurate hand detection. This modification guarantees more reliable feature extraction, especially for small or difficult-to-detect hand features that might be missed with a shallower network structure.

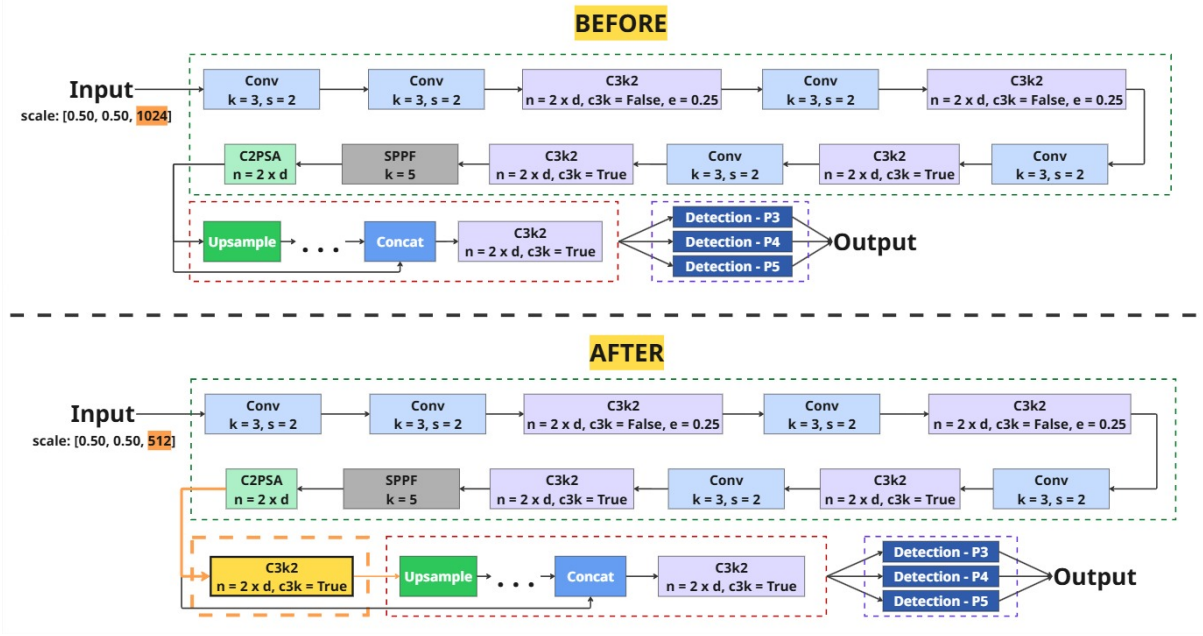


Figure 3.8: Visual comparison between the base YOLO11 architecture and our tweaks.

3.4 Model Training

3.4.1 Training Process

In the first step, the VSL dataset that we have prepared and preprocessed will be downloaded from the Roboflow library (see Algorithm 1).

Algorithm 1 Download Dataset from Roboflow

1: Install the Roboflow library
2: Import Roboflow module
3: Initialize Roboflow with API key
4: Access the workspace with the given name
5: Select the project named "vsl-words"
6: Choose version of the project
7: Download the dataset in YOLOv11 format

Before implementing the custom dataset to YOLO model, we will need to create a YAML file. YAML is a human-readable data format used to configure various aspects of YOLO. The YAML file helps the YOLO model configure the data path, specifies the location of the training images and their corresponding labels (bounding box information contains the class labels). This tells the model where to find the data it needs to train on. In addition to this, the YAML can also define the model architecture, including the number of convolutional layers, filters, activation functions, etc. Although the YAML file is automatically generated and attached to the downloaded dataset, we still need to rewrite it to match the class names in Vietnamese and configure the data path since we use Google Colab as the training environment. See Figure 3.9 to see the YAML file we use in this study to configure the data path and name the labels.

```yaml
path: ../VSL-Words-1
train: train/images
val: valid/images
test: test/images

nc: 20
names: ['Ăn', 'Bạn', 'Bố', 'Cảm ơn', 'Có', 'Đói', 'Giúp', 'Khát', 'Không', 'Không có gì', 'Là gì?', 'Mẹ', 'Tạm biệt', 'Tên', 'Tôi', 'Uống', 'Việt Nam', 'Xin chào', 'Xin lỗi', 'Yêu']

roboflow:
  workspace: toan-nguyen-hcq45
  project: vsl-words
  version: 1
  license: Public Domain
  url: https://universe.roboflow.com/toan-nguyen-hcq45/vsl-words/dataset/1
```

Figure 3.9: Example of YAML file.

3.4.2 Loss Function

The loss function used in our YOLO model for hand sign detection combines several different components to improve both the accuracy of finding objects and classifying them correctly. This approach helps the model learn more effectively by addressing different aspects of the detection task simultaneously. We used three main types of loss functions in our model. First, we implemented Complete Intersection over Union (CIoU) loss for bounding box regression. This component appears as "box_loss" in our training records and helps improve how accurately the model can locate objects by making sure the predicted boxes align well with the actual object locations. Second, we incorporated Distribution Focal Loss (DFL), which shows up as "dfl_loss" in our training logs. This component helps the model handle uncertainty better when detecting objects and improves how well it can estimate which category an object belongs to. Third, we used Varifocal Loss (VFL) as part of our classification loss, shown as "cls_loss". This component specifically addresses problems that occur when some types of hand signs appear much more frequently than others in the training data. It also helps handle uncertainty in classification tasks, ensuring that our model can reliably identify different hand signs.

The training process for YOLO models involves many different settings called hyperparameter that control how the optimization works. We paid special attention to configuring the loss function components properly.

Two important parameters directly affect how much weight each loss component receives during training. The "box" parameter controls the weight given to bounding box loss, which determines how much the model focuses on predicting precise coordinates. The "cls" parameter controls the weight of classification loss, which determines how much emphasis the model places on making accurate class predictions. These weights are crucial because they help balance the trade-off between localization (finding where objects are) and classification (identifying what objects are). By adjusting these weights, we can make the model focus more on the aspects that are most important for our specific detection task.

After conducting extensive experiments and fine-tuning, we adjusted the loss function weights to better match the goals of hand sign detection. Our task focuses on detecting single hand signs, where only one sign appears in each image at a time. Since our task does not involve the complex challenge of locating multiple objects in crowded scenes, we reduced the box weight from 7.5 to 5. This change reflects a decreased emphasis on extremely precise bounding box coordinates, since hand signs typically occupy a significant portion of the image and do not require as precise localization as small objects in complex scenes. On the other hand, we increased the cls_loss's weight from 0.5 to 1, which emphasizes our study's main objective of accurately classifying different hand signs. Classification accuracy, the most important component of our application, is given priority in this modification. These changes maximize the model's performance by prioritizing

classification accuracy while preserving enough localization capabilities. The particular requirements of hand sign identification, where accurately identifying the type of sign is more crucial than obtaining pixel-perfect bounding box coordinates.

The total loss for the proposed YOLO model is defined as:

$$\begin{aligned} L_{\text{total}} = \lambda_{\text{box}} \cdot \sum_{i=1}^{N} (1 - \text{CIoU}_i) \\ + \lambda_{\text{obj}} \cdot \sum_{i=1}^{M} [-y_i \log(p_i) - (1 - y_i) \log(1 - p_i)] \\ + \lambda_{\text{cls}} \cdot \sum_{i=1}^{N} \sum_{c=1}^{C} \left[-c_{i,c}^{gt} \log(c_{i,c}) - (1 - c_{i,c}^{gt}) \log(1 - c_{i,c})\right] \end{aligned}$$

Where the loss weights are:

$$\lambda_{\text{box}} = 5, \quad \lambda_{\text{obj}} = 1.5, \quad \lambda_{\text{cls}} = 1$$

Explanation of Parameters:

- N: Number of positive samples (anchors matched with ground-truth boxes).
- M: Total number of objectness predictions (across all grid cells and anchors).
- C: Number of object classes.
- CIoU_i: Complete Intersection over Union for the i-th matched bounding box, which includes overlap, center distance, and aspect ratio penalty.
- p_i: Predicted objectness score for the i-th anchor (probability that an object exists).
- $y_i \in \{0, 1\}$: Ground truth objectness label (1 if an object exists, 0 otherwise).
- $c_{i,c}$: Predicted probability of class c for the i-th bounding box.
- $c_{i,c}^{gt} \in \{0, 1\}$: Ground truth class label for class c of bounding box i.
- λ_{box}: Weight for the bounding box regression loss (set to 5 from 7.5).
- λ_{obj}: Weight for the objectness loss (set to 1.5 as default).
- λ_{cls}: Weight for the classification loss (set to 1 from 0.5).

3.4.3 Evaluation Metrics

The performance of deep learning models for hand signal detection is evaluated using several key metrics, including True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN), along with the Intersection over Union (IoU) function. The True Positive indicates that the model correctly detected the hand signs from the expected bounding box, which is inside the actual bounding box. A False Negative happens when the model fails to detect a hand signs present in the ground truth, and a True Negative represents correctly identifying images without hand signs. Since there are no images in the collection without hand signs, the measurement is not considered in our analysis. The precision, recall, F1 score, and mAP values of the system are taken into consideration

in order to evaluate the performance of the model trained with all of these metric data. Based on these metrics, the model's effectiveness is further assessed using precision, recall, F1 score, and mean average precision (mAP), which collectively measure the accuracy and completeness of the detections, with the IoU providing additional insight into how well the predicted bounding boxes align with the actual hand signs.

- **Precision:** Measures the accuracy of positive predictions made by the model. It defines how many predicted as positive were correctly predicted. Precision value is calculated as:
$$Precision = \frac{TP}{TP + FP}$$
- **Recall:** The percentage of favorable outcomes that were accurately predicted. Recall evaluates how well the model captures all real positive examples. Even if the model occasionally misses some positive thoughts, high recall suggests that it can detect them successfully. Recall formula:
$$Recall = \frac{TP}{TP + FN}$$
- **F1-Score:** Calculates the harmonic weight of the precision and recall values, providing a balanced measure of a model's overall performance. It aims to find a compromise between precision and recall. The formula to get the value of F1-score:
$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$
- **mAP (or Mean Average Precision):** Uses to measure the performance of a model for tasks such as object detection tasks and information retrieval. The average precision (AP) value is a popular metric for object detection detectors. Calculation of the mAP value:
$$mAP = \frac{1}{N} \sum_{k=1}^{k=N} AP_k$$
(with N = the number of class; AP_k = the average precision of class k)

Besides the mentioned metrics, there are some popular units used when talking about measurement in real-time application: milliseconds (ms) is known as the most common measurement in inference, FPS or Frame Per Second is the number of images that can be processed per second. In real-time applications, the models are required to process the data in milliseconds with low extraction rate or high FPS.

3.4.4 Training Configurations

Beside the augmentations we applied when creating the dataset, during the training process, YOLO also add some augmentations setting as the hyperparameter to prevent overfitting during training:

- **Mosaic:** Combines four training images into one, simulating different scene compositions and object interactions.

- **Mix-up:** Blends two images and their labels, creating a composite image. Enhances the model's ability to generalize by introducing label noise and visual variability.

  → This helps the model learn to recognize objects in many different types of scenes and lighting conditions. When the model sees these mixed images during training, it gets better at understanding complex situations where objects might look different than usual.

- **Erasing:** Randomly erases a portion of the image during classification training, encouraging the model to focus on less obvious features for recognition.

- **Translate:** Translates the image horizontally and vertically by a fraction of the image size, aiding in learning to detect partially visible objects.

  → This forces the model to learn to recognize objects even when only small parts are visible. The model has to focus on finding other clues and features that might not be obvious at first.

  Figure 3.10 shows a sample of the data augmentation processing during training process. (a) is an original image in Sign Language Letters Dataset. (b) results from Mixup processing. (c) is the result of processing using Cutout. (d) is the result of processing using Erasing.

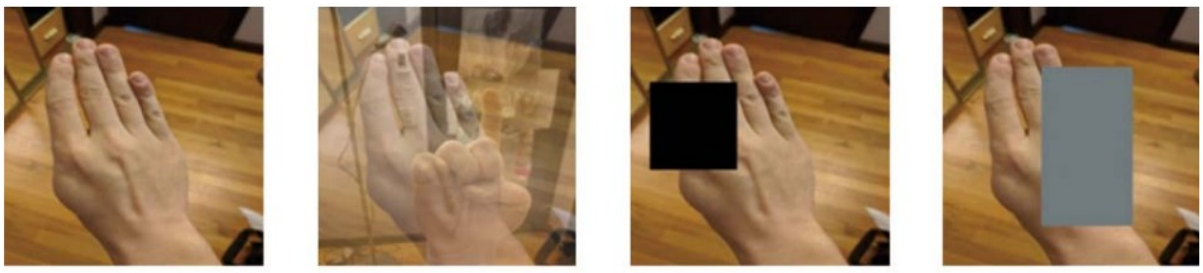


Figure 3.10: Sample of the YOLO augmentation process. [31]

To ensure the best performance of our proposed YOLO model, we experimented with multiple optimization algorithms and finally come up with the SGD (Stochastic Gradient Descent) as the optimizer [32]. SGD is a default optimizer used for deep learning models, Gradient Descent is an iterative optimization process that searches for an objective function's optimum value (Minimum/Maximum). This is one of the most used methods for changing a model's parameters in order to reduce a cost function in machine learning projects.

Because the raw images come in different sizes, we want to resize them to the same size before parsing them into the YOLO model later. More specifically, we want both the length and width to be 640 pixels.

The model was trained using our proposed YOLO architecture with the following settings:

Table 3.3: Training Configuration

Model	Proposed YOLO
Epochs	200
Batch	32
Workers	8
Optimizer	SGD
Image size	(640, 640)

3.5 Postprocessing

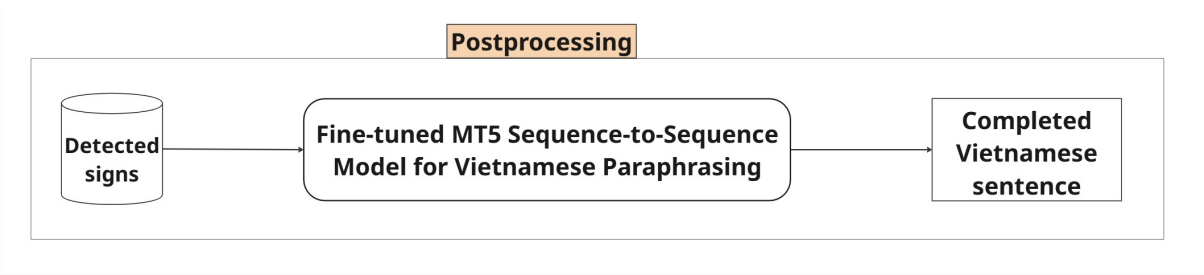


Figure 3.11: Postprocessing pipeline.

After predicting, the output will go through a Fine-tuned MT5 Sequence-to-Sequence Model from Hugging Face [33] to paraphrase and generate a completed sentence from the predicted words, as illustrated in Figure 3.11. Hugging Face is an open-source library that provides access to thousands of pre-trained transformer models, state-of-the-art ML models, and datasets for various natural language processing (NLP) and computer vision tasks such as text classification, question answering, translation, generation and so forth [34]. A transformer model is a type of neural network that tracks relationships in sequential input, such as the words in a sentence, to understand context and meaning. Transformer models identify subtle ways that even distant data items in a series influence and depend on one another by using a growing collection of mathematical procedures known as attention or self-attention. First described in a 2017 research paper named "Attention Is All You Need" from Google [35], transformers are among the newest and one of the most powerful classes of models invented to date. They're driving a wave of advances in machine learning some have dubbed transformer AI [36].

The model **vietnamese-sentence-paraphrase** is a tool designed specifically for paraphrasing Vietnamese sentences [37]. It leverages the MT5 architecture, a multilingual variant of the T5 model [38], to generate paraphrased versions of input Vietnamese text. This model use a paraphrase dataset in English from Hugging Face [39] and translate it to Vietnamese. Users can easily run the model by loading its tokenizer and conditional generation model from the Hugging Face Transformers library, then feeding sentences into the model to receive paraphrased outputs.

Currently, this model can only be implemented in our app for processing uploaded files, not for real-time detection. This limitation arises because real-time detection typically uses WebSocket connections, which require very low latency and high responsiveness. Integrating a Transformer-based model like this one directly into a WebSocket-based real-time pipeline tends to significantly decrease performance and slow down the app. Transformer models, including the MT5-based paraphrase model, are computationally intensive and

introduce noticeable processing delays. When used in real-time over WebSocket, these delays can leading to a slower user experience and increased latency beyond what WebSocket communication itself would add. While WebSocket protocols are optimized for fast, continuous data exchange with minimal overhead (with round-trip times in the tens of milliseconds), the heavy computation of the Transformer model cannot match this speed, causing the app response slowly or become unresponsive [40]. Therefore, restricting the model's use in the processing of uploaded files avoids these real-time performance issues, ensuring smoother operation.

CHAPTER 4

IMPLEMENTATION AND RESULTS

This section provides a concise summary of the experimental results for the proposed YOLO model, including their interpretation, the resulting experimental conclusions, a comprehensive assessment of key changes and improvements compared to the original YOLO11 model, and a comparison with previous studies.

4.1 Implementation

The model is trained on Google Colab as a virtual machine to make use the Tesla T4 GPU (see Figure 4.1). With the help of GPU, our training process can be accelerated to be done in a much faster way.

```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| 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   38C    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 4.1: Tesla T4 GPU Configuration.

The preprocessed dataset is divided into 5137 images (90%), 286 (5%), and 285 (5%) for training, validation, and testing, respectively among 20 word classes in VSL. Figure 4.2 presents the number of images obtained in each class, showing a balanced dataset.

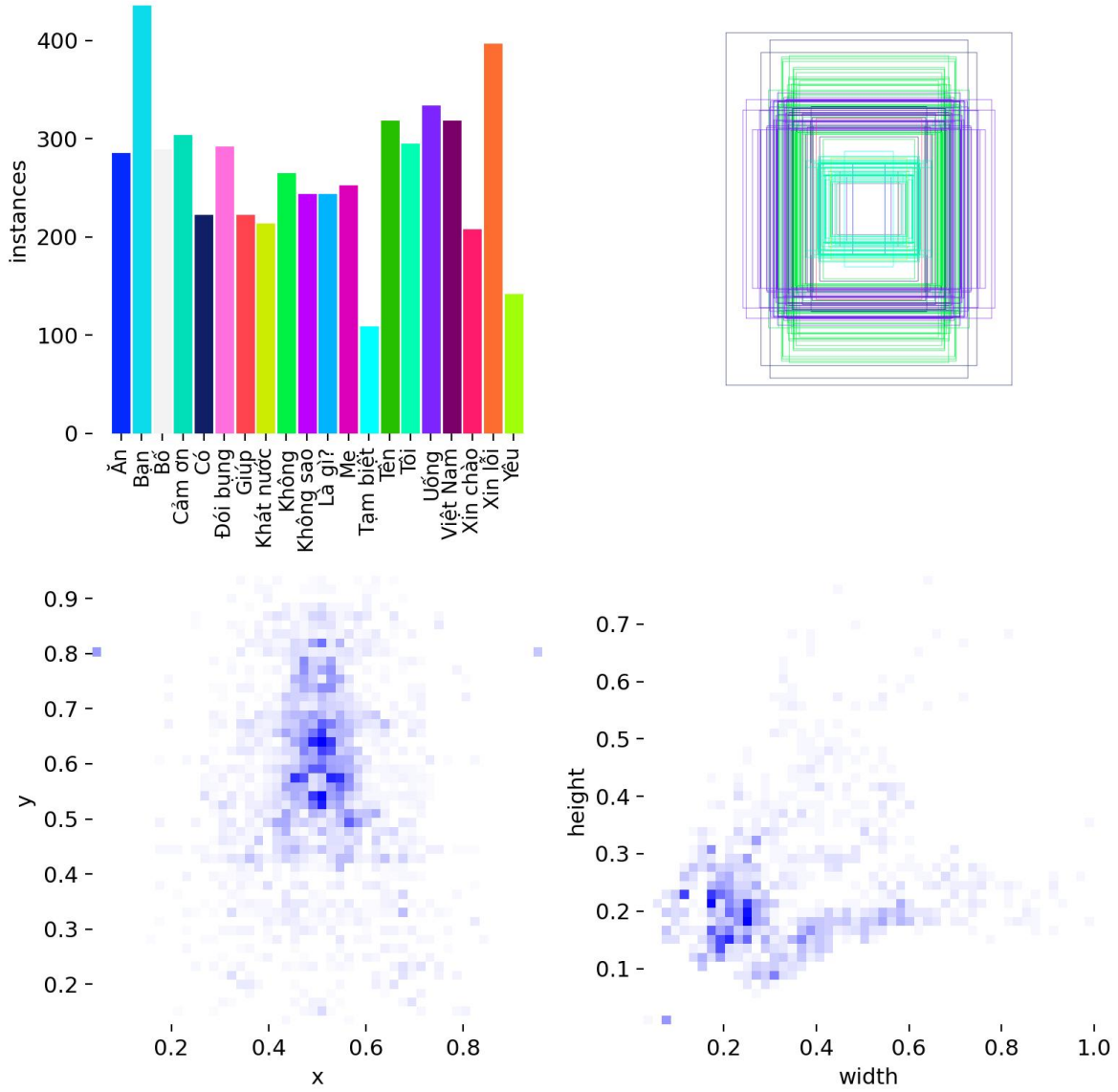


Figure 4.2: Classes Distribution of the VSL Dataset.

The training process takes about 7 hours to finish training for 200 epochs. After training, an average time for recognizing is 0.2ms preprocess and 3.0ms postprocess per image or per frame in a video sequence. The inference time will be depend on the device the model was run on, for the Colab environment, the average inference time is 2.8ms. Therefore, the total processing time per image is 6ms.

4.2 Experimental Outcomes

In Figure 4.3, the training and validation box loss, class loss, object loss, precision metrics, recall metrics and the mAP consecutively for the trained model is represented. In general, the more epoch we train, the better bounding box prediction and the better predicted box object categorization.

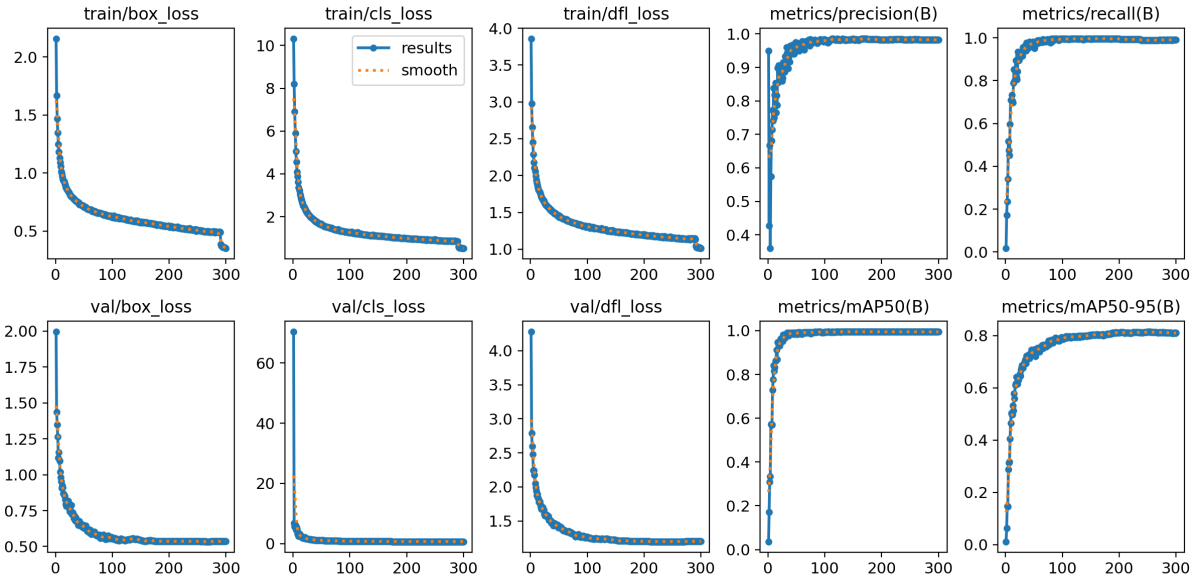


Figure 4.3: Final Result.

The confusion matrix of the proposed model has been presented in Figure 4.4 and Figure 4.5, illustrate the performance of a classification model across 20 distinct classes. In general, we see mostly 1.00, which means the model nailed it for most of the phrases. However, for some classes like class "Tôi" shows a slightly lower accuracy of 92%, with 8% of its instances misclassified as "Tạm biệt" or the misclassifications of "Không có gì" into "Là gì".

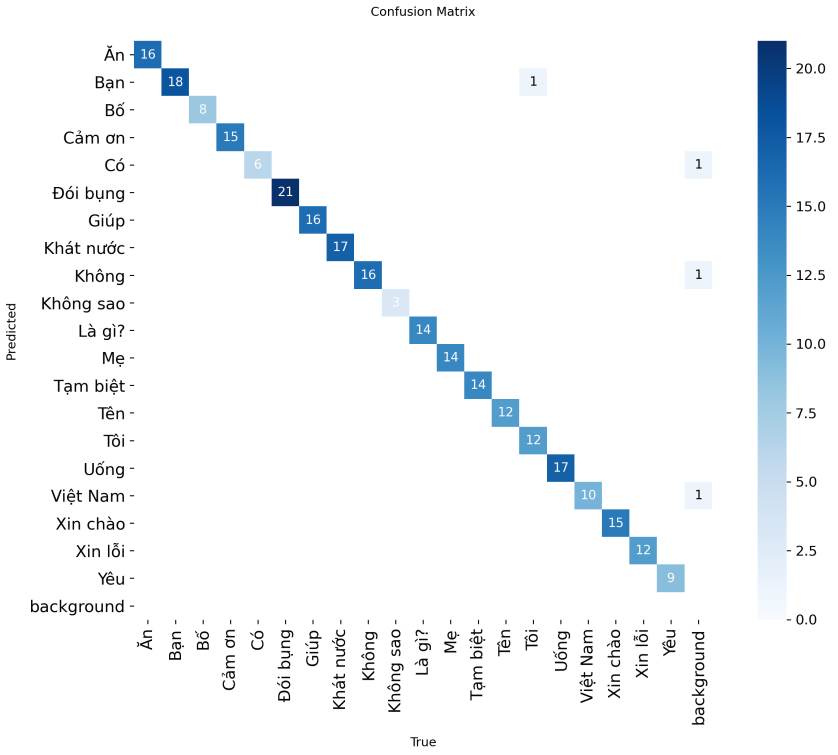


Figure 4.4: Confusion Matrix.

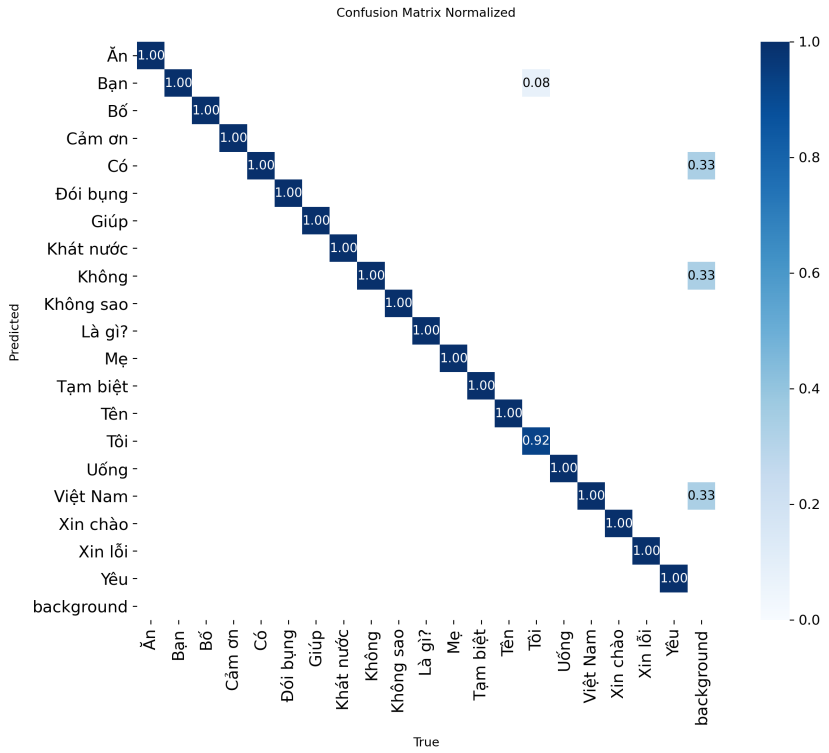


Figure 4.5: Confusion Matrix After Normalized.

In Figure 4.6, the graph's y-axis displays the F1 score, and the x-axis displays confidence levels. The model's confidence curve appears to be 67.0%, indicating that the peak F1-score for all classes combined approaches 0.99 when the model's confidence threshold is set at about 0.670. This high score indicates that the model is performing very successfully.

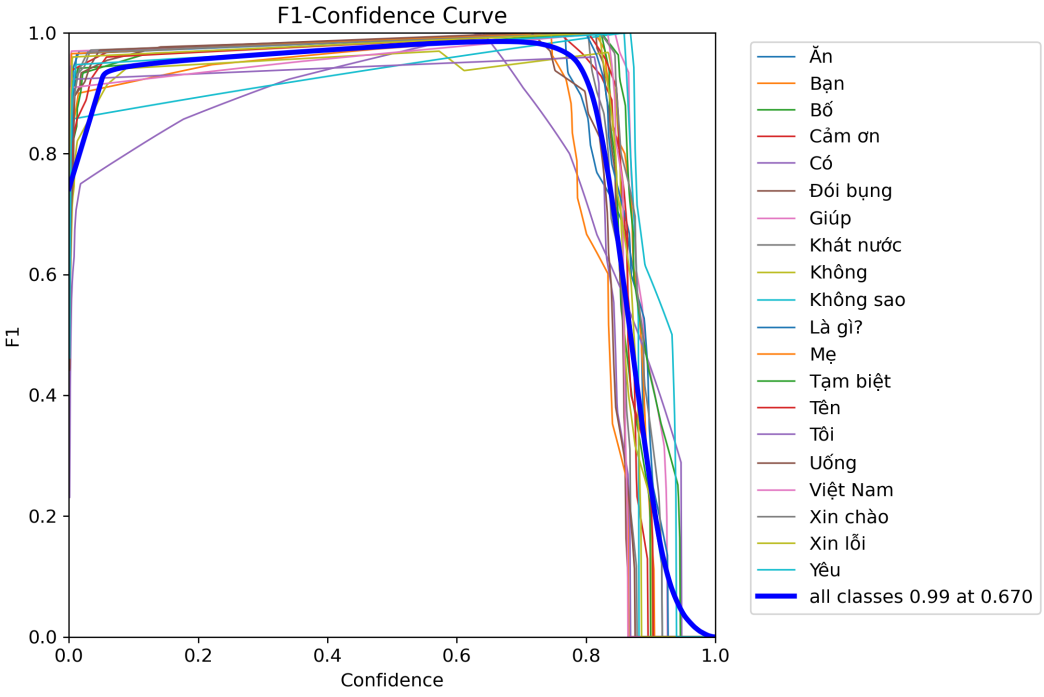


Figure 4.6: F1-Confidence Curve.

In Figure 4.7, the x-axis displays the model's estimated probability of forecast accuracy, or the confidence threshold. Model accuracy at each confidence level is displayed on the y-axis. When the model forecasts 100% precisely, the bold blue line displays the precision of 0.919 (91.9%) for all classes. This model is perfect since it makes accurate and certain predictions. High confidence thresholds, however, have the potential to reduce recall by causing the model to overlook a large number of genuine positives when it lacks the confidence to predict them.

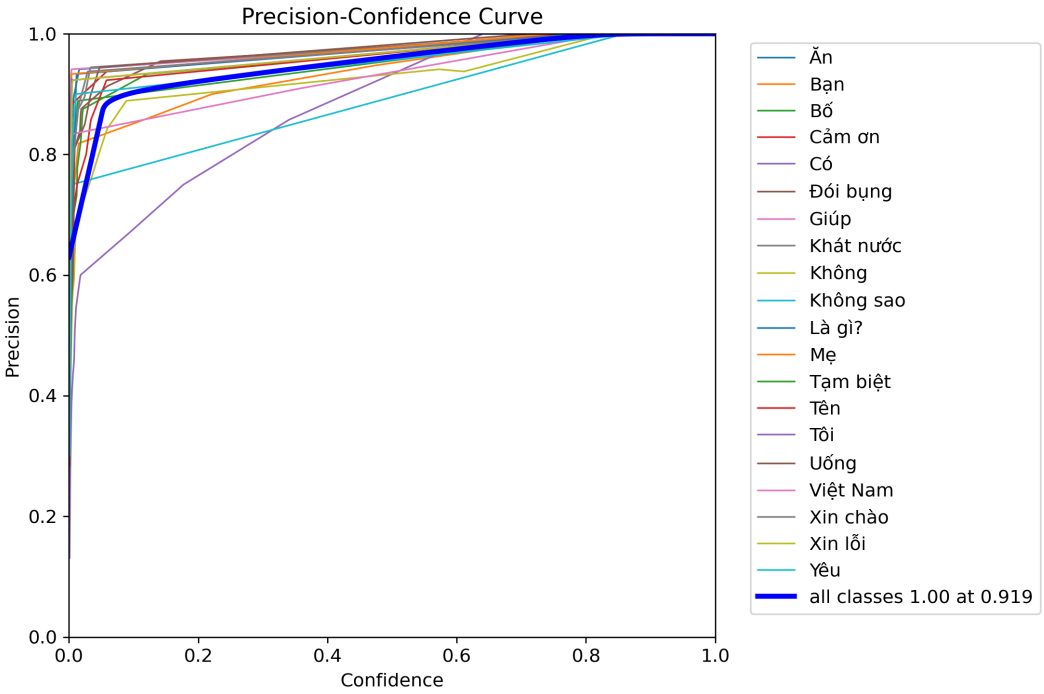


Figure 4.7: Precision-Confidence Curve.

In Figure 4.8, recall is represented by the x-axis, and precision by the y-axis. Many models aim to achieve a position close to the upper-right quadrant of the graph by optimizing both precision and recall. A mean Average Precision (mAP) of 0.995 at an IoU (Intersection over Union) threshold of 0.5, a frequently used criterion in object detection tasks, is achieved by the model, as indicated by the blue line, "all classes 0.995 mAP@0.5". In addition to this, according to Figure 4.3, the mAP50-95 indicates the Mean Average Precision calculated at varying IoU thresholds is approximately of 81%. The mapped value is noticeably high, indicating that the model performs well at this threshold in terms of both precision and recall across all classes. However, randomly dividing video frames between train and validation sets will produce artificially high mAP and poor generalization in the real world. Moreover, since the dataset includes self-recorded and photographed, these results are not highly reliable in practice.

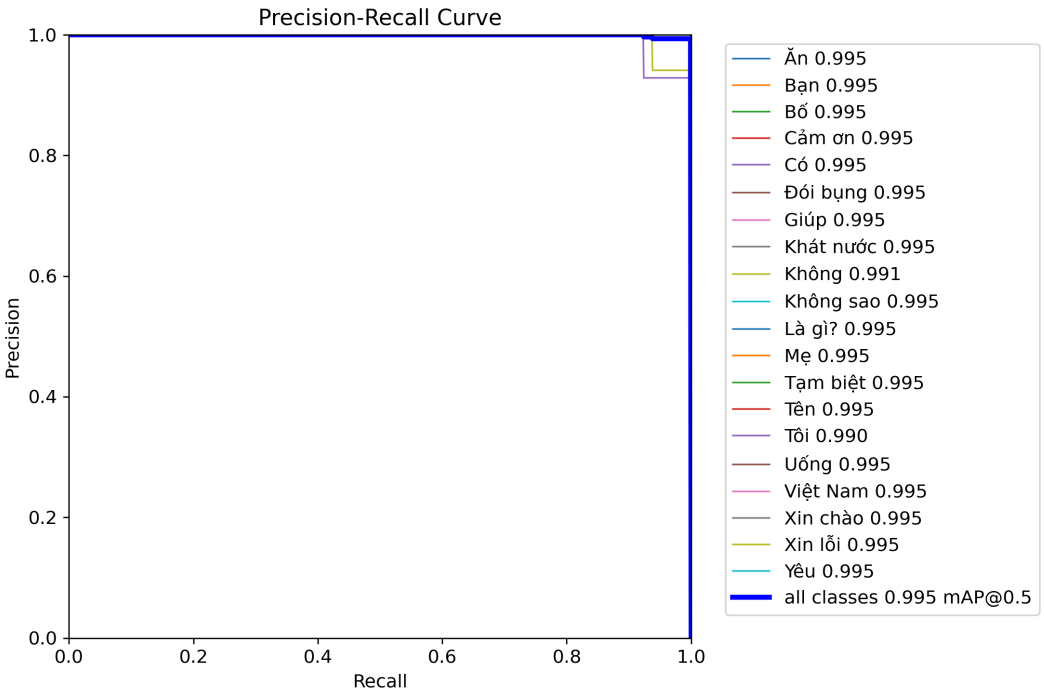


Figure 4.8: Precision-Recall Curve.

Figure 4.9 displays the predicted model's detection performance for different signs in the validation set. The findings demonstrate that the model has a high confidence for the majority of the indications. Accuracy, precision, and recall are all robustly demonstrated by the suggested detection model, which also has the lowest training and validation loss.

Figure 4.9: Prediction on Validation Set.

4.3 Key Changes and Improvements

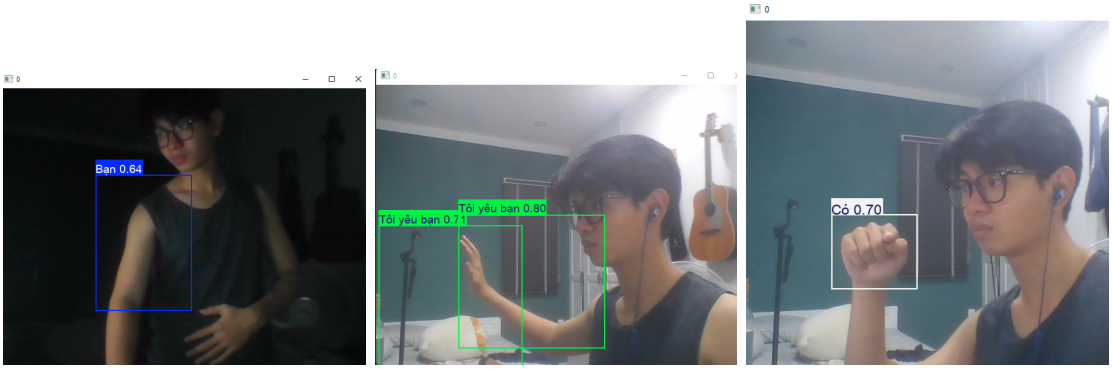
The new YOLO model we developed shows major improvements compared to the original YOLO11 model after training for 200 epochs on our VSL dataset, the details are shown in Table 4.1. Most importantly, our model needs much fewer computer resources to run. We reduce the GPU memory usage by 36%, going down from about 7.97GB to 5.36GB and takes up less storage space (shrinking from 19.2MB to only 6.8MB). These changes make the model more suitable for real-time and resource-constrained environments without sacrificing performance and also work faster when processing images. In addition to this, the time it takes to analyze each image dropped from 5.1ms to 2.8ms, and the cleanup work afterward went from 5.1ms to 3.0ms. While training the new model does take a bit longer (about 7 hours instead of 6 hours), the results are much better. The model became more accurate at finding objects, with the recall score improving from 0.984 to 0.993. This means it catches more of the things it's supposed to find. The precision stayed excellent at 1.00, showing it rarely makes mistakes when it does detect something. The F1 score, which combines both accuracy measures, also got slightly better. Overall, these improvements make our proposed YOLO model more efficient and effective. It's now better suited for real-world use because it delivers high accuracy while requiring less

computing power and resources.

Table 4.1: Comparison of Original YOLO11 Model vs Proposed YOLO Model

	Original YOLO11 Model	Proposed YOLO11 Model
GPU memory	~7.97G	~5.36G
Model Size	19.2MB	6.8MB
Processing Time (per image)	• 0.2ms preprocess • 5.1ms inference • 0.0ms loss • 5.1ms postprocess	• 0.2ms preprocess • 3.1ms inference • 0.0ms loss • 3.0ms postprocess
Training Time	~6 hours	~7 hours
Recall	0.984	0.995
Precision - Confidence	1.00 at 0.988	1.00 at 0.919
F1 - Confidence	0.97 at 0.766	0.99 at 0.670

The original YOLO11 model has some problems when trying to recognize hand gestures in real-time video (see Figure 4.10). The main issues are that it often misses gestures that are actually there, and it sometimes thinks it sees gestures when there aren't any. These problems get much worse when the lighting is poor or dark, causing the system to work very badly in low-light situations. When there isn't enough light, the YOLO11 model struggles because it can't see the details it needs to tell the difference between a real gesture and just random hand movements. The camera picks up blurry or unclear images in the dark, making it hard for the computer to understand what's happening. Since the model was trained mostly on clear, well-lit pictures, it doesn't know how to handle dark or shadowy conditions very well.



(a) False detection in dark (b) False Detection (c) Miss detection

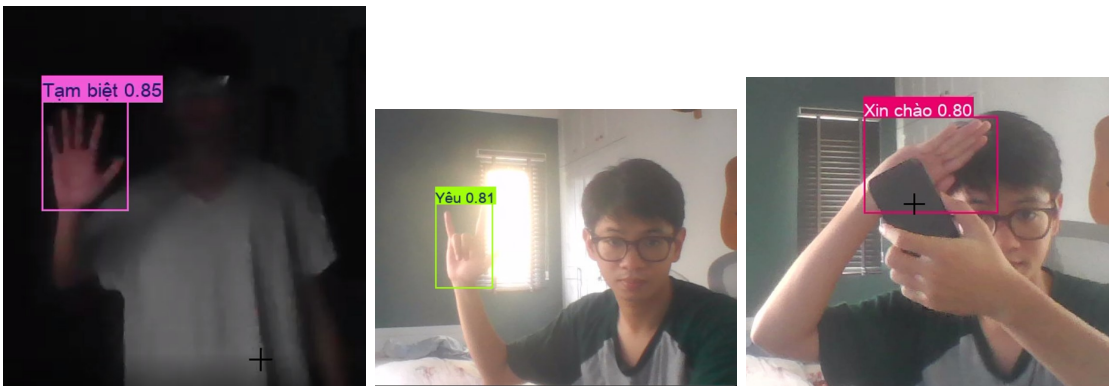
Figure 4.10: Problems of original YOLO11 model.

This creates big problems for people who want to use gesture control systems in the real world. Many places don't have perfect lighting all the time - like rooms with dim lights, outdoor areas at night, or spaces where you can't control how bright it is. When the gesture recognition doesn't work properly in these common situations, it becomes frustrating and unreliable for users. This means the technology can't be used in many practical applications where people would actually want gesture control to work, limiting how useful the YOLO11 system really is for everyday use.

To solve these problems, we have developed an image augmentation framework that not only artificially increases the size of available datasets in scenarios of limited training

datasets, makes the dataset more visibly diverse , but also improves the detection ability of applied object detectors in different. By learning from the blended images from Mosaic and Mix-up, the model becomes better at recognizing objects even when the lighting is poor, the hand sign is against the light, or the image is noisy and unclear. Erasing and Translate teach the model to find and recognize objects that are only partially visible in the picture frame. When objects are cut off at the edges or partly hidden, the trained model can still detect them because it learned to work with incomplete views during training.

After testing 3 common problems a detector can facing in real-time environment, The results can be seen in Figure 4.11.



(a) Low lighting (b) Silhouette (c) Occlusion

Figure 4.11: Results of data augmentation in real-time environment.

4.4 Comparison with Prior Studies

To explain the study's improvement, we compare our findings to current studies on Vietnamese sign language recognition systems, which we discussed in Chapter 2, in terms of accuracy and efficiency. However, each study use the different dataset (except study on 2019 and 2021 use the same dataset [8][9]). Therefore, given an overall comparison and value judgment on the precise and accuracy among all these studies is almost impossible.

4.4.1 Accuracy

According to several researchers, accuracy is a requirement metric for evaluate a success of a deep learning model [6][9]. However, despite its widespread use and importance, accuracy as a standalone metric presents significant limitations when attempting to assess a system's overall performance capabilities. When accuracy is compared against other evaluation measures, it becomes evident that relying solely on this metric fails to provide a complete and nuanced understanding of how well a system actually performs in real-world scenarios. The F1 score is a much better way to measure performance because it looks at two important things at once: precision (how many correct results we get) and recall (how many actual results we find). Similarly, the mean Average Precision (mAP) represents another crucial metric because it checks two things at the same time: location accuracy (how well the system draws boxes around objects it finds) and classification accuracy (how well the system identifies what those objects actually are). These metrics are particularly valuable because they capture aspects of system performance that accuracy alone cannot

represent. In many real applications, knowing exactly where an object is located is just as important as knowing what it is.

Given these considerations and the need for comprehensive performance evaluation, our research uses several different measurements together, including the F1 score, mAP values, and other performance indicators to thoroughly test our system from multiple angles. Table 4.2 summarize the results of all research on VSL detection models from 2017 to 2024 with our proposed model.

Table 4.2: Performance Comparison of Proposed YOLO Model with Prior VSL Studies

Model	**Accuracy (%)**	**mAP@0.5**	**Precision (%)**	**Recall (%)**	**F1 (%)**
Proposed YOLO Model	-	**99.3**	**91.9**	**99.5**	**99**
SVM (2017)[6]	80–95	-	-	-	-
VGG16 and LSTM (2019)[8]	95.83	-	-	-	-
GoogLeNet and BiLSTM (2021)[9]	99.38	-	-	-	-
Heatmap and Depth (2024)[10]	80.15	82.5	-	-	-
Bio-impedance (CNN) (2024)[11]	95.12	-	-	-	-

Our experimental results demonstrate exceptional performance across these comprehensive evaluation criteria. With a mean Average Precision (mAP) of 0.993 at an IoU (Intersection over Union) threshold of 0.5, our model performs almost perfectly at identifying objects. This outstanding result reflects the successful integration of both localization and classification accuracy within our system architecture. The high mAP value indicates that our model not only correctly identifies and classifies objects within images but also accurately predicts their spatial boundaries through precise bounding box estimation. A previous study in 2021 had an accuracy of 99.38% on validation and 98.15% on test data, demonstrating its capacity to accurately classify feature sequences [9]. These results showed that the system of the study in 2021 was very good at classifying objects correctly. However, these accuracy numbers only tell us about classification performance - they don't tell us anything about how well the system located objects in images. This difference in evaluation creates a problem when trying to compare our current research with the previous study. The earlier study's reliance on accuracy-based metrics means that detailed insights regarding localization performance remain unavailable, making it difficult to assess how well that system performed in terms of spatial precision and bounding box prediction accuracy.

As a result, without the inner sight information regarding localization, it's difficult to make a fair comparison between the previous work results and our current research.

4.4.2 Efficiency

For the efficiency of the models, we will take the detection speed into consideration. Table 4.3 show the analysis of image processing per image or frame revealed by the

following time distribution and the requirements to achieve that processing time.

Table 4.3: Efficiency and Requirement of Proposed YOLO Model vs. Prior Works

Model	**Processing Time**	**Requirements/Notes**
Proposed YOLO Model	**6ms per frame**	Standard webcam only
GoogLeNet + BiLSTM	5–7 seconds for 3-second video	Slow pre-processing and classification
Bio-impedance	400ms per gesture	Requires physical sensors and low generalization
Heatmap + Depth	51.11ms per frame	Needs depth sensors

In the study when the authors use the BiLSTM model for the training stage, the average time for recognizing a video is 4-5s for preprocess and 1-2s for classifying a video with the frame rate is 30 Frame per Second (FPS) and the video length is 3 seconds, in overall it took about 5-7s for detected a 3s video. This evaluation was done by Matlab R2019a environment with CPU Intel Core™ i5-3210M-2.5GHz, 8 GB of RAM [9]. On the other hands, the total time our model will take to classify is approximately 6 seconds per frame in the video when testing on a laptop environment with CPU AMD Ryzen 7 5700U with Radeon Graphics, 8 GB of RAM. Based on this information, we can conclude that, our proposed YOLO model shows much better potential for real-time applications. With processing times approximately of 1.0ms preprocessing and 5.1ms postprocessing per frame on our device, our model could potentially handle live video streams more effectively than the previous CNN and BiLSTM approach. The Multi-channel Bioimpedance-based Device which was created in 2024 takes 400ms to detect a gesture [11], offers portability, cost-effectiveness, robustness to environmental conditions (like lighting), improved privacy, and lower power/computing requirements compared to camera-based systems. However, our proposed YOLO model is significantly faster for real-time detection (6ms per frame compared to 400ms per gesture) and does not require wearing any physical sensors or electrodes, offering greater user convenience. Moreover, our model also process much more faster than the combination of heatmap and depth approach in 2024 (6ms per frame compare to 51.11ms per frame) [10].

In addition to this, our model doesn't need any extra equipment like the AcceleGlove in [4] or the Kinect camera in [6] to detect a gesture, the signer only uses a webcam to record their hand signs.

4.5 Web Application Development

For the accessibility of our YOLO-based VSL detection model to everyone, we create an application called the VSL Translator, a powerful application with a friendly UI design. Built with a React frontend, it offers a smooth and intuitive interface for users to upload media, including images or videos, and view real-time detection results through device's Webcam. The FastAPI backend ensures a high-performance web framework for building our proposed YOLO model APIs, efficient processing, delivering accurate Vietnamese Sign Language interpretations. This tool is ideal for educators, researchers, and the deaf community to engage with VSL effortlessly.

4.5.1 Building the Backend

The first step in backend server development involves choosing an suitable framework, enabling developers to efficiently build and integrate features. In this project, FastAPI (Figure 4.12) is selected as the core framework for developing the backend server of web applications, particularly those involving machine learning model deployment. Using asynchronous programming, the modern, high-performance Python web framework FastAPI is incredibly effective at managing multiple requests at once. FastAPI makes it easier to create scalable and maintainable APIs by providing a wide range of HTTP methods and middleware features. It allows the automatic creation of interactive API documentation using ReDoc and Swagger UI, which significantly improves API usability and developer experience. All of these characteristics work together to create reliable, easy-to-use APIs that are quick, simple to create, and maintain. For backend servers that need to integrate with our proposed YOLO model and complex web application features, FastAPI is a great option.

Figure 4.12: FastAPI Framework. [41]

For this project, we used Uvicorn as the ASGI server to run the FastAPI backend. Uvicorn is known for its high performance and efficient handling of asynchronous tasks, making it well-suited for modern Python web frameworks like FastAPI. By leveraging Uvicorn, the backend can efficiently manage multiple concurrent requests, which is essential for applications involving real-time inference and API interactions. And also, this help our application become deployable in the near future.

The Figure 4.13, as shown in the Swagger UI documentation, exposes three main API endpoints, each serving a specific purpose in the system workflow.

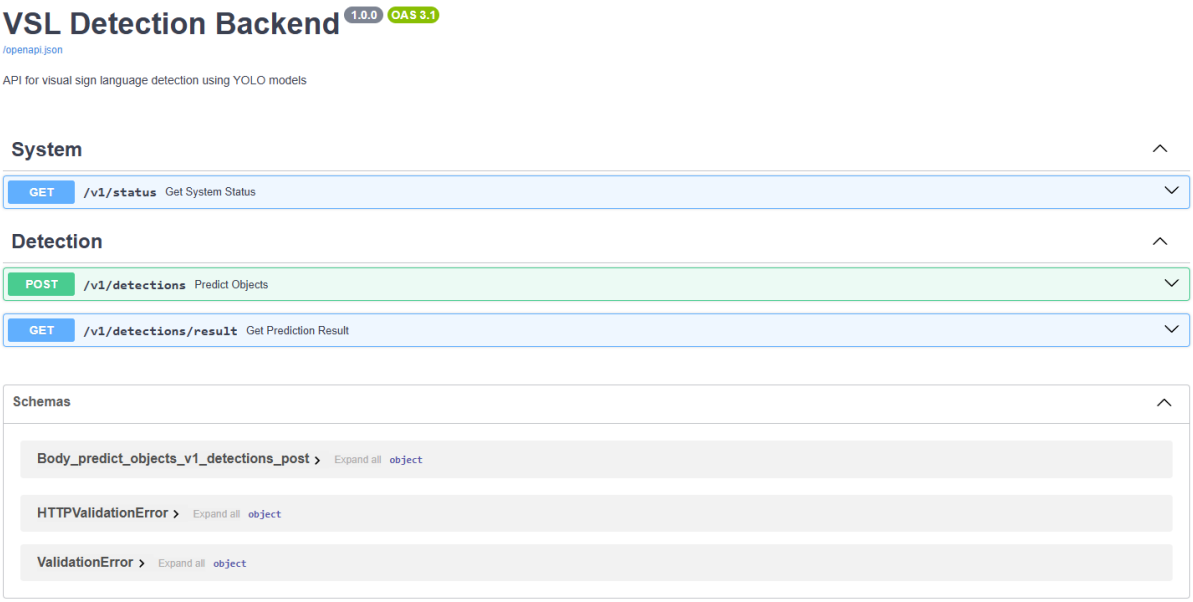


Figure 4.13: API endpoints.

GET /v1/status

- **Function:** This endpoint is used to check the system status.
- **Description:** It returns information about the current state of the backend server.

POST /v1/detections

- **Function:** This endpoint is responsible for predicting objects.
- **Description:** Clients can send data, such as images or video frames, to this endpoint. The backend processes the input using the YOLO model to perform visual sign language detection. The result of the prediction is then returned to the client. This is the core endpoint for running inference tasks.

GET /v1/detections/result

- **Function:** This endpoint provides the prediction result.
- **Description:** After a detection request is made, clients can use this endpoint to get the result of the previous prediction. This allows for asynchronous processing, where the client can submit a request and then poll for the result once it is ready.

WebSocket /v1/detections/stream

- **Function:** Real-time detection via continuous video streaming.
- **Description:** Establishes a WebSocket connection for bidirectional communication. Clients stream video frames or sequences to the server in real time. The server processes each frame using the YOLO model and instantly returns detection results.

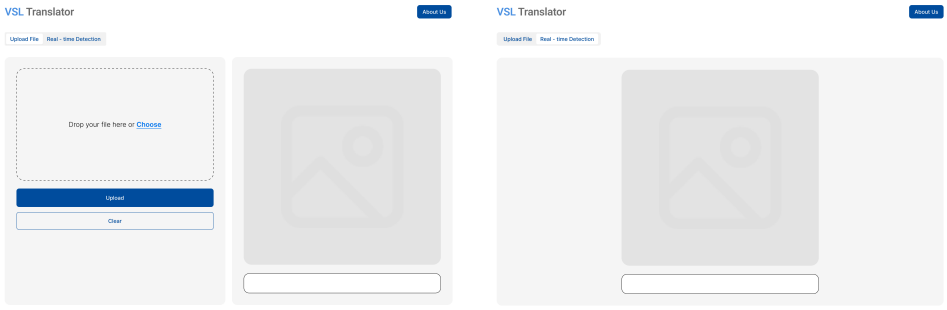
By combining HTTP and WebSocket APIs, the backend supports both batch processing and real-time use cases, making it versatile for diverse applications in visual sign language detection.

4.5.2 Building the Frontend

In this project, the User Interface (UI) has been developed using ReactJS (See Figure 4.14). React is the library for web and native user interfaces. Build user interfaces out of individual pieces called components written in JavaScript [42]. Its component-based design, performance optimization, and rich interactivity enable the creation of a responsive and user-centric interface that enhances the overall effectiveness of the Vietnamese sign language recognition application. React's ability to support large-scale apps and potential integration with React Native for mobile platforms offers future scalability.

Figure 4.14: React Library.

The design for the application is show in Figure 4.15 using Figma. As we planned, there will be 2 main pages in the application, the first page is for file uploading, users can drop the video or image having the sign language in the left column and the translation result will be show in the right column. The other page will be used for real-time detection, the application will access to your device's webcam, translate the hand signs and show the result in real-time.



(a) Page 1 (b) Page 2

Figure 4.15: The original design of the VSL Translator

4.5.3 Integrating and Testing

After finalizing the design, the next step is to implement the user interface (UI), translating the visual and functional plans into actual code. This implementation phase is crucial because it shapes how users will interact with the application. Therefore, we can see the final product UI will be quite different from the blueprint, with more detail, more focus on the visual and interactive, especially the real-time detection page. Once the UI is developed, it must be integrated with the backend, to ensure smooth and coherent functionality. Following integration, testing is performed to verify that all components

work together correctly and meet the intended requirements. This section discusses how the UI was implemented and polished based on the design and describes the integration and testing processes that ensure the web application operates effectively and reliably.

Figure 4.16 is the final UI for the File-uploader page when it is in the idle state. Users will drop or upload the file in the left column, the website currently accepts 4 extensions: **.mov** and **.mp4** for video files, **.jpg** and **.png** for image files. Then the file's information including the file name, the file size, and the file type will be show under the drop area. The users now can press the "Upload" button to start detecting.

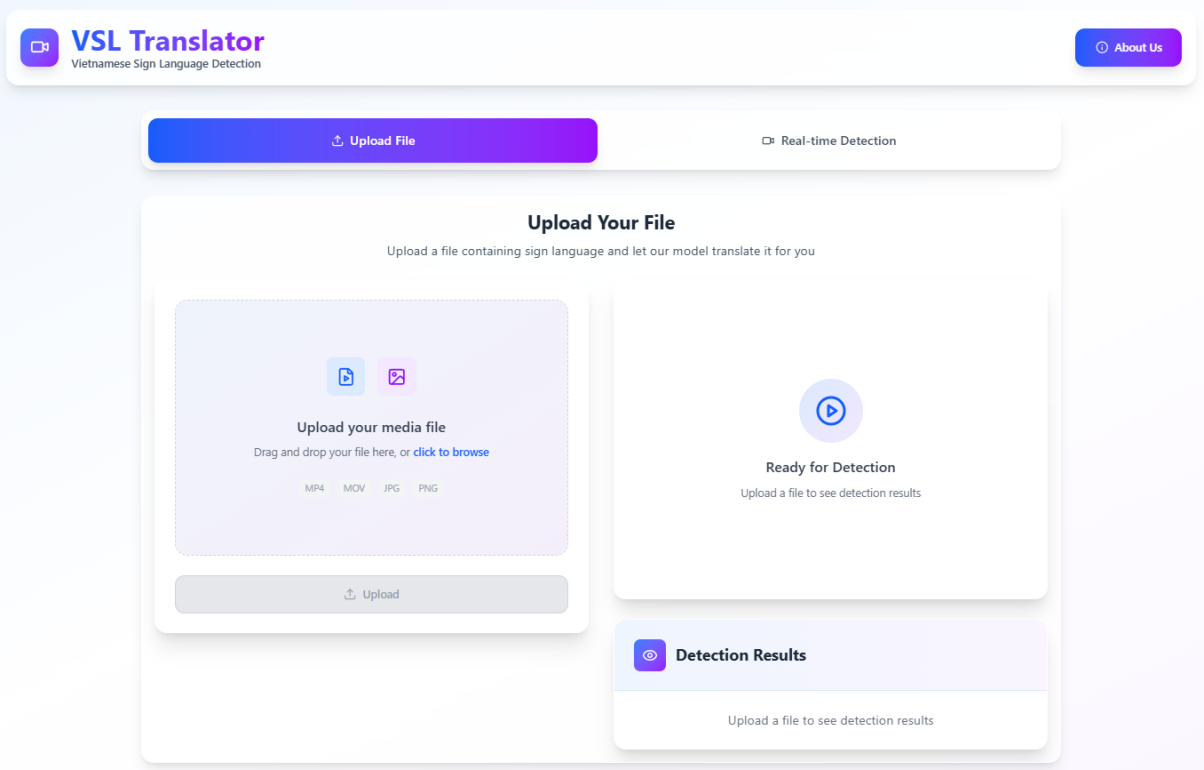


Figure 4.16: File-upload Page.

When the user upload an image or a video the result will be show in the right column with the predicted class as well as the confidence below the "Detection Results" area. There will also have an area to show the sentence created using the detected words below the detected words. The examples for image upload and video upload are showing in Figure 4.17 and Figure 4.18 consecutively.

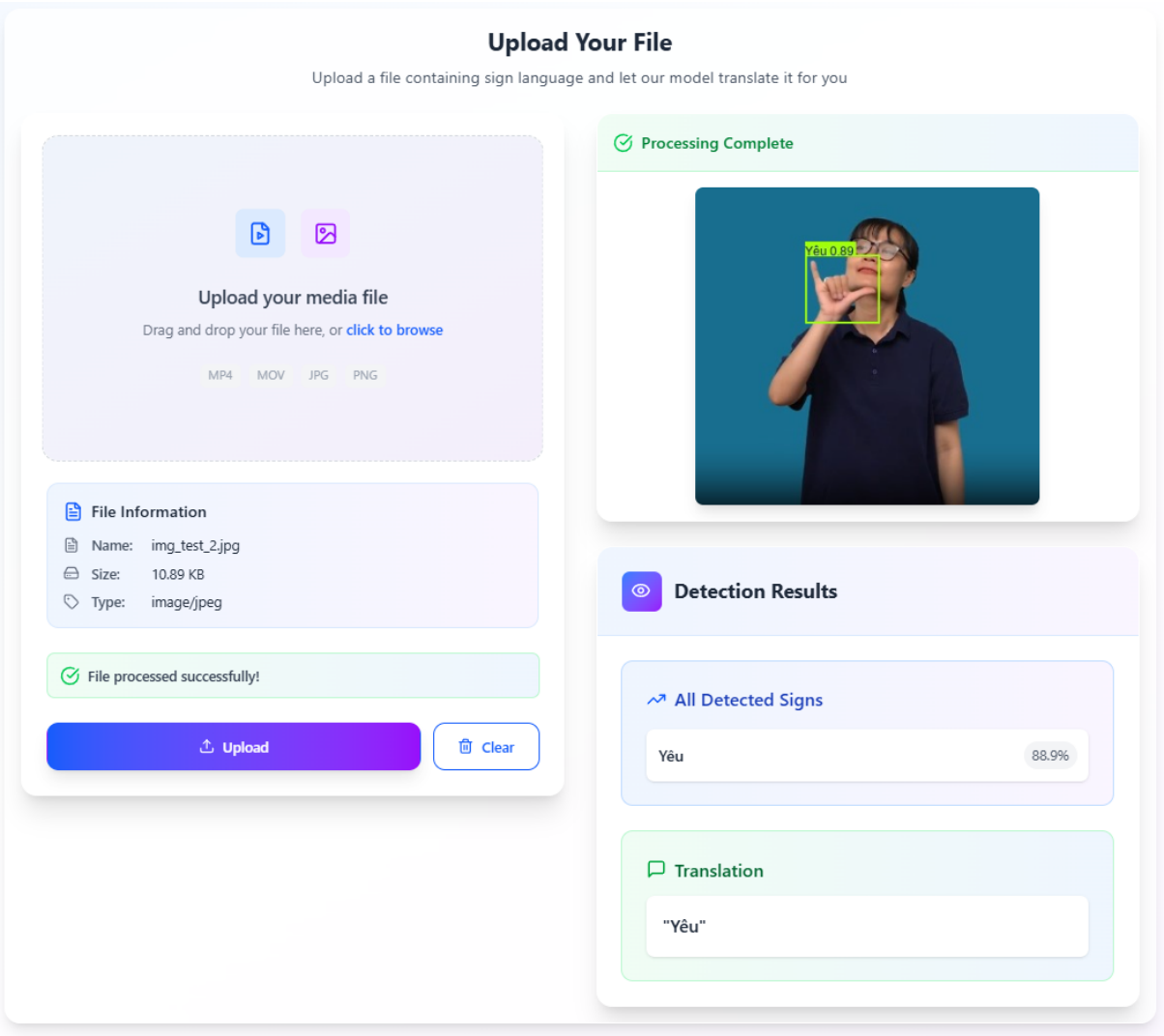


Figure 4.17: Example of image upload.

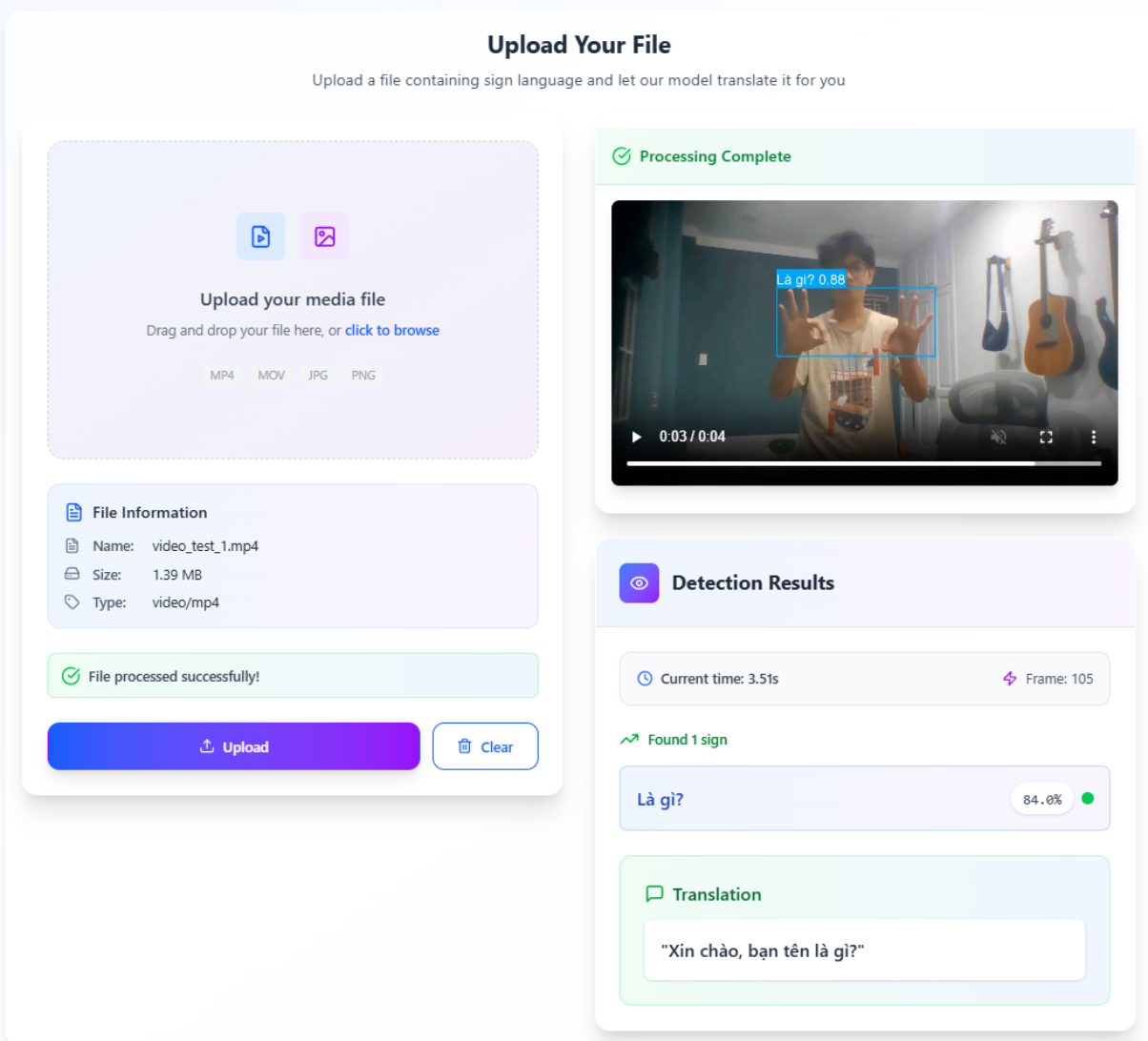


Figure 4.18: Example of video upload.

Moving to the Real-time detection tab, we will have the column in the left shows the current camera. The user can also choose to turn on or turn off the show processed frames with detection boxes. When the user is ready, his/her can press the "Start Real-time Detection" button to start real-time detecting. The results as well as the predicted classes will be continuously updated and displayed in the right column in real-time. For more details, see Figure 4.19.

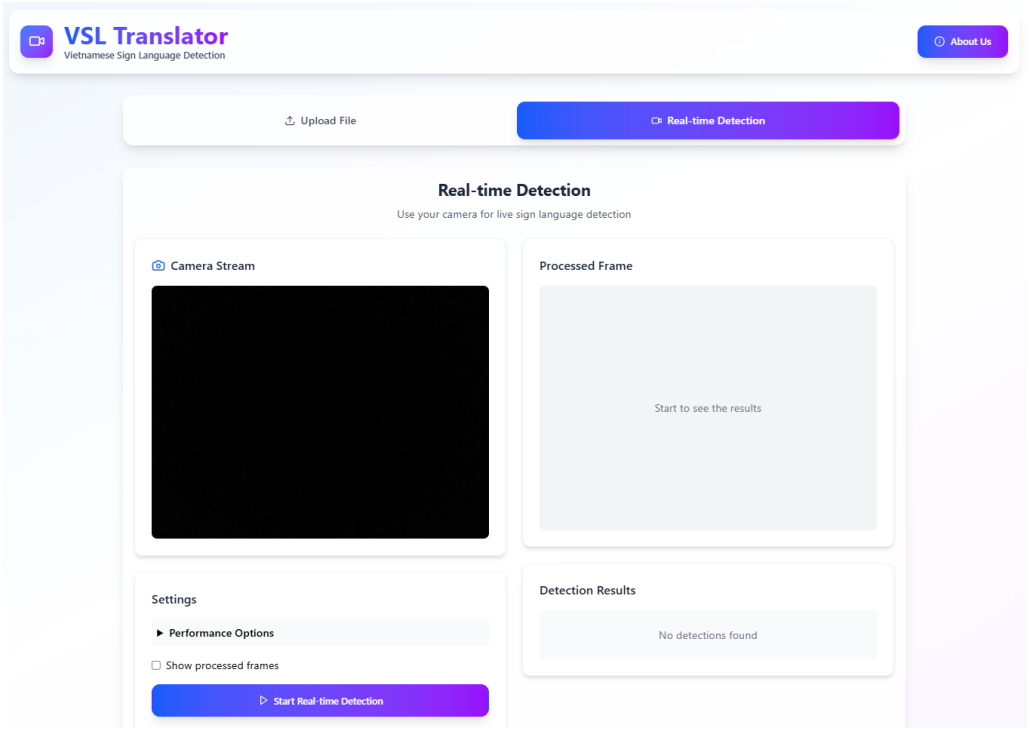


Figure 4.19: Real-time Detection Page.

Figure 4.20 shows an example of a frame during real-time detection.

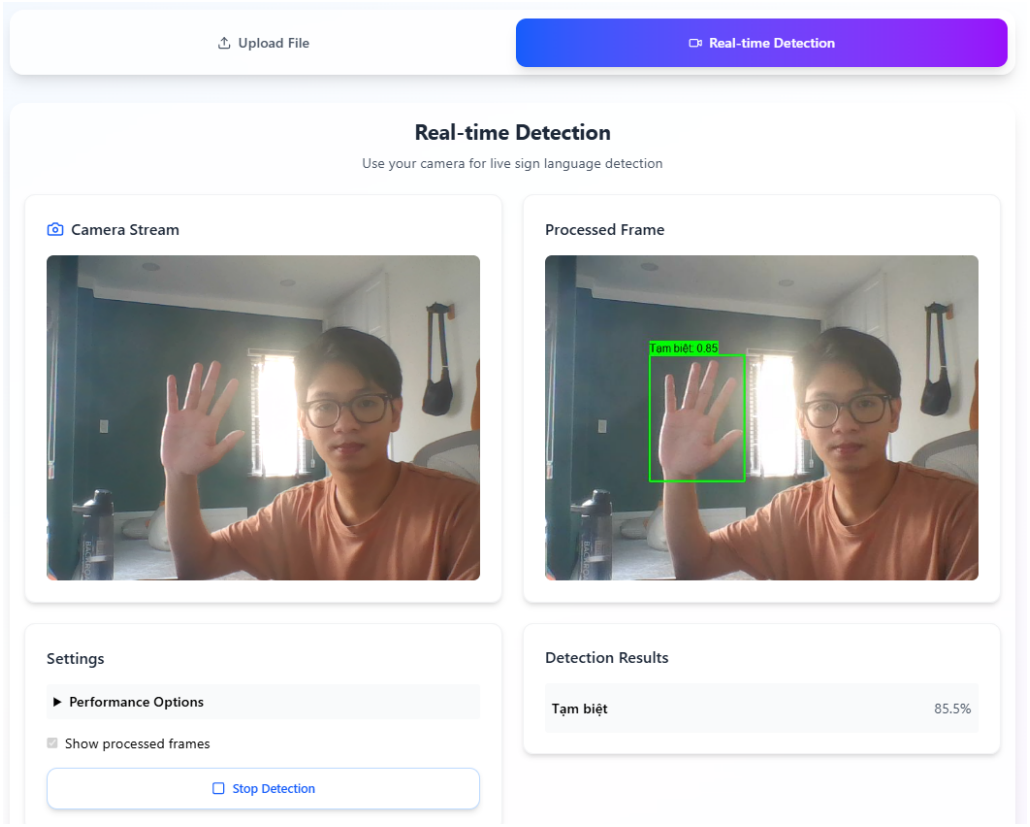


Figure 4.20: Example of real-time detection.

CHAPTER 5

CONCLUSION

5.1 Conclusion

In this paper, we propose a novel detector for real-time Vietnamese sign language recognition, which is based on YOLO11 with improvements of the backbone, neck, and data augmentation. The proposed model is optimized by reducing channels from 1024 to 512 and using smaller initial filters to lower complexity while maintaining essential features for better generalization and hand detection accuracy. An added C3k2 block after the attention module refines attention-enhanced features, and increasing the C3k2 block depth from 2 to 3 layers improves multi-scale feature extraction and small hand detection. These changes result in a lighter, faster model with enhanced precision and efficiency. The final result has produced exceptional performance metrics, such as a mean Average Precision (mAP) of 99.5%, mAP50-95 is 81%, a precision of 91.9%, a recall rate of 99.5%, and a F1 score of 99%. These metrics validate the effectiveness of our proposed YOLO model for precise hand gesture detection by showing the model's ability to detect and classify VSL movements with minimum errors. As a result, we think this study will serve as a cutting-edge guide for future investigations.

However, it is essential to acknowledge the limitations of this work. Due to the lack of VSL dataset in the training resource, one notable constraint lies in the light and background factors. These factors can affect tremendously on the performance of the model, decrease the accuracy and cause false detections, which may pose challenges when deploy the application in public. Additionally, it is necessary to recognize another limitations of this work during the data preparation process is that the entire sign language recognition is carried out only for the manual component and with static sign language pictures as input, as well as the data labeling has to be done manually by hand which takes a large amount of time. Therefore, a new method should be innovated to deal with the data input and data labeling. The performance of the proposed model needs to be improved in terms of faster processing of more information and practical applications by enrich the diversity in the existing dataset.

5.2 Future Work

In the future, we intend to expand the dataset to include a wider range of hand forms and motions, as well as more words from the VSL vocabulary, in order to recognize entire phrases. This modification aims to increase the model's capacity so that gesture recognition can practically promote the communication and interaction between people with disabilities and ordinary people, and can be better used in the fields of disability assistive devices, medical and healthcare. Moreover to this, one limitation our model should be noticed is that the current dataset appears limited in scope, covering only a subset of VSL vocabulary in hand forms. This restricts the model's ability to recognize the full complexity of sign language communication, which includes not just individual signs but also grammatical structures, facial expressions, and contextual variations. The speed of processing uploaded files is another limitation that should be take into consideration. Currently in our application, users are only allowed to upload one file at once and the

processing time of the YOLO model when handle a video is quite slow since it detect frame by frame. Therefore, in the future we would like to upgrade the site to handle multiple files at once and find ways to speed up the video processing of the YOLO model. Furthermore, we would like to enhance the model so that it can increase the accuracy and the efficiency in real-time environment. Our proposed model has provided some solutions to handle practical problems such as lighting and occlusion. However, the accuracy when testing on the above conditions is still not high enough to apply in practice. For real-time detecting and paraphrasing, we intend to develop an alternative lightweight model or server-side asynchronous processing to maintain responsiveness without degrading the WebSocket communication performance. Last but not least, the model may struggle with maintaining consistent frame rates while preserving accuracy, particularly on resource-constrained devices. Therefore, we will continue to develop and deploy an app version of this model available to all devices to make it easily accessible to all individuals of the deaf and hard-of-hearing community.

In conclusion, these advancements will be a significant step forward in the development of assistive technology for the Vietnamese community, enabling more equitable communication and convenience in daily life.

REFERENCES

[1] World Health Organization WHO, "Deafness and hearing loss," 2 Feb. 2024. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[2] G. Jocher and J. Qiu, "Ultralytics YOLO11." [Online]. Available: https://docs.ultralytics.com/models/yolo11/

[3] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[4] T. D. Bui and L. T. Nguyen, "Recognizing Postures in Vietnamese Sign Language with MEMS Accelerometers," *Sensors Journal, IEEE*, vol. 7, pp. 707 – 712, Jun. 2007.

[5] Hernandez-Rebollar, "Talking glove speaks for the deaf," Aug. 2003. [Online]. Available: https://www.cbsnews.com/news/talking-glove-speaks-for-the-deaf/

[6] D. H. Vo, H. H. Huynh, D. Mien, and J. Meunier, "Dynamic Gesture Classification for Vietnamese Sign Language Recognition," *International Journal of Advanced Computer Science and Applications*, vol. 8, Mar. 2017.

[7] C. D. Mutto, P. Zanuttigh, and G. M. Cortelazzo, *Microsoft Kinect Range Camera*. Boston, MA: Springer US, 2012, pp. 33 – 47.

[8] H. V. Anh, P. Van-Huy, and T. N. Bao, "Deep Learning for Vietnamese Sign Language Recognition in Video Sequence," *International Journal of Machine Learning and Computing*, vol. 9, Aug. 2019.

[9] Q. P. Van and B. N. Thanh, "Vietnamese sign language recognition using dynamic object extraction and deep learning," in *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*, 2021, pp. 402–407.

[10] X.-P. Nguyen, T.-H. Nguyen, D.-T. Tran, T.-S. Bui, and V.-T. Nguyen, "An isolated vietnamese sign language recognition method using a fusion of heatmap and depth information based on convolutional neural networks," in *2024 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2024, pp. 1–6.

[11] N.-M. Than, S.-T. Nguyen, D.-N. Huynh, T.-N. Tran, N.-K. Le, H.-X. Mai, C.-D. Le, T.-T. Pham, Q.-L. Huynh, and T.-H. Nguyen, "A multi-channel bioimpedance-based device for vietnamese hand gesture recognition," *Scientific Reports*, vol. 14, no. 1, p. 31830, 2024.

[12] H. Pham The, H. Chau, V.-P. Bui, and K. Ha, "Automatic feature extraction for vietnamese sign language recognition using support vector machine," *2018 2nd International Conference on Recent Advances in Signal Processing, Telecommunications and Computing (SigTelCom)*, pp. 146–151, Jan. 2018.

[13] C. Börstell, "Lexical comprehension within and across sign languages of Belgium, China and the Netherlands," *Glossa: a journal of general linguistics*, vol. 8, May 2023.

[14] [Online]. Available: https://qipedc.moet.gov.vn/

[15] "Roboflow: Computer vision tools for developers and enterprises." [Online]. Available: https://roboflow.com/

[16] "Roboflow Annotate: Label Images Faster Than Ever." [Online]. Available: https://roboflow.com/annotate

[17] N. Joseph, "The importance of blur as an image augmentation technique," Mar 2020. [Online]. Available: https://blog.roboflow.com/using-blur-in-computer-vision-preprocessing/

[18] N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, "Evaluating the evolution of yolo (you only look once) models: A comprehensive benchmark study of yolo11 and its predecessors," *arXiv preprint arXiv:2411.00201*, 2024.

[19] A. Imran, M. S. Hulikal, and H. A. Gardi, "Real time american sign language detection using yolo-v9," *arXiv preprint arXiv:2407.17950*, 2024.

[20] "Turi Create simplifies the development of custom machine learning models." [Online]. Available: https://apple.github.io/turicreate/docs/userguide/object_detection/

[21] A. Alsharif, R. Shanmugam, M. Masud, F. Alzahrani, and J. Arshad, "Yolov8-based american sign language alphabet detection using mediapipe and transfer learning," *Franklin Open*, vol. 1, p. 100013, 2024.

[22] H. Jia and Y. Li, "Slr-yolo: An improved yolo-based model for real-time sign language recognition," *Journal of Intelligent & Fuzzy Systems*, vol. 46, no. 5, pp. 4893–4903, 2024.

[23] M. Bhuiyan, N. Nahar, I. Hossain, and L. Nahar, "American sign language phrase detection using yolov5 and yolov8," 2024, arXiv preprint.

[24] A. Alsharif, M. Masud, F. Alzahrani, J. Arshad, and R. Shanmugam, "Sign language alphabet detection using yolov11 and mediapipe for enhanced accuracy," *Sensors*, vol. 25, no. 6, p. 2501, 2025.

[25] T. Navin, S. Das, S. Akter, and M. H. Rony, "Detection of bangla and english sign alphabet using yolov11 model," *Vision*, vol. 9, no. 2, p. 22, 2025.

[26] M. Imran, A. Khan, S. Rahman, and J. Alam, "Yolov9 based real-time american sign language detection using transfer learning," 2024, arXiv preprint.

[27] F. Ahmed, T. Rahman, T. Hossain, and N. Alam, "Bangla sign language detection using yolov10 model," in *Proceedings of TCCE 2025: Emerging Trends in Computing and Communication*, ser. Lecture Notes in Networks and Systems, vol. 841. Springer, 2025, pp. 512–523.

[28] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "Yolov8 to yolo11: A comprehensive architecture in-depth comparative review," *arXiv preprint arXiv:2501.13400*, 2025.

[29] "GitHub - ultralytics/ultralytics: Ultralytics YOLO11." [Online]. Available: https://github.com/ultralytics/ultralytics

[30] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," *arXiv preprint arXiv:2410.17725*, Oct. 2024.

[31] W. Jia and C. Li, "SLR-YOLO: An improved YOLOv8 network for real-time sign language recognition," *Journal of Intelligent & Fuzzy Systems*, vol. 46, no. 1, pp. 1663–1680, 2024.

[32] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[33] [Online]. Available: https://huggingface.co/

[34] IBM, "What is hugging face?" IBM, 2025. [Online]. Available: https://www.ibm.com/think/topics/hugging-face

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[36] R. Merritt, "What is a transformer model?" Mar. 2022. [Online]. Available: https://blogs.nvidia.com/blog/what-is-a-transformer-model/

[37] C. N. Quang, "Vietnamese sentence paraphrase model (chieunq/vietnamese-sentence-paraphase)," 2023. [Online]. Available: https://huggingface.co/chieunq/vietnamese-sentence-paraphase

[38] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mT5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[39] V. Vorobev and M. Kuznetsov, "Chatgpt paraphrases dataset (humarin/chatgpt-paraphrases)," 2023. [Online]. Available: https://huggingface.co/datasets/humarin/chatgpt-paraphrases

[40] M. Hlayel, H. Mahdin, and H. A. Mohd Adam, "Latency analysis of websocket and industrial protocols in real-time digital twin integration," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 73, no. 1, pp. 120–135, January 2025. [Online]. Available: https://ijettjournal.org/Volume-73/Issue-1/IJETT-V73I1P110.pdf

[41] S. Ramírez, "FastAPI," 2018. [Online]. Available: https://fastapi.tiangolo.com/

[42] J. Walke, "React," 2013. [Online]. Available: https://react.dev/