Jurnal Nasional Teknik Elektro dan Teknologi Informasi
Volume 14 Number 4 November 2025


DOI: 10.22146/jnteti.v14i4.21931

# Application of You Only Look Once (YOLO) Method for Sign Language Identification

**Reni Triyaningsih[1], Pradita Eko Prasetyo Utomo[2], Benedika Ferdian Hutabarat[2]**

[1] Information Systems Study Program, Faculty of Science and Technology, Jambi University, Muaro Jambi, Jambi 36361, Indonesia
[2] Informatics Study Program, Faculty of Science and Technology, Jambi University, Muaro Jambi, Jambi 36361, Indonesia

[Submitted: 2 July 2025, Revised: 5 October 2025, Accepted: 30 October 2025]
Corresponding Author: Pradita Eko Prasetyo Utomo (email: pradita.eko@unja.ac.id)

**ABSTRACT** — Limited understanding of sign language has widened the social gap for deaf people, creating barriers in communication and social interaction. To address this challenge, technology-based solutions are required to facilitate inclusive communication. Deep learning-based detection methods, particularly the You Only Look Once (YOLO) algorithm, have gained attention for their speed and accuracy in real-time object detection. This research aims to develop and evaluate a YOLO training model for the identification of Indonesian sign language system (*sistem isyarat bahasa Indonesia*, SIBI). The dataset was obtained from resource person at the State Special School Prof. Dr. Sri Soedewi Masjchun Sofwan, SH. Jambi, and enriched with additional images collected from external subjects. Augmentation techniques with Roboflow were applied to expand the dataset, and several training schemes were implemented. Model performance was assessed using confusion matrix while considering accuracy and indications of overfitting. The results showed that the quality and quantity of training data, as well as the epoch values, strongly influenced the accuracy of the trained model. The best performance was achieved with 40 primary images per label class, augmented to 60 images, and trained over 24 epochs, resulting in a confusion matrix accuracy of 99.9%. The implemented model was able to recognize SIBI gestures in real-time using a webcam with fast processing. Overall, the proposed YOLO-based model successfully identifies sign language in real-time and demonstrates strong potential for reducing communication barriers among deaf people. However, further refinement and expansion of the dataset are recommended to improve effectiveness and enable broader real-world applications.

**KEYWORDS** — Sign Language, Real-Time Detection, You Only Look Once, YOLO Model Fine-Tuning.

## I. INTRODUCTION

Sign language is a form of communication commonly used by deaf people using hand gestures, facial expressions, and body movements to form symbols that represent letters or words [1]. Just like any other language, sign language is a naturally evolving system, systematic, and governed by linguistic rules [2]. In Indonesia itself, the formal use of sign language as a means of communication and teaching material that has been formalized by the government is the Indonesian sign language system (*sistem isyarat bahasa Indonesia*, SIBI) [3]. SIBI is a form of oral communication adapted to sign language, with some vocabulary taken from American sign language (ASL) [4]. SIBI was developed through a combination of four local, forged, invented, and absorbed signs, which were then standardized into a national sign system. It is published by the Ministry of Education and Culture and applied in formal schools, such as special schools [5]. The purpose of using SIBI in education is to help people with disabilities actively participate, especially in teaching and learning activities, to improve the quality of their social interactions. Hearing and communication impairments will not prevent them from doing their usual activities, including continuing their education. They can still communicate using sign language.

In Indonesia, sign language is generally learned through special schools and the deaf community, as well as specialized books that discuss sign language. Implementing learning activities in a deaf community is certainly not easy and is very different from learning in a nondisabled environment. For deaf people, they must receive education guided by qualified teaching staff [6]. However, the nondisabled environments generally learn without the need for sign language teaching staff, so they become less familiar with sign language.

Limited understanding of sign language among nondisabled individuals further widens the social gap between them and the deaf community. This situation create barriers for deaf people in social interaction, emotions, and communication [7]. Communication can be said to be successful if what is conveyed and intended by someone can be understood by the interlocutor [8]. However, deaf people are more likely to have difficulty communicating with the nondisabled community. As a result, they are often viewed differently by the nondisabled community, causing them to feel inferior and hopeless.

To realize quick action to overcome the communication gap between deaf people and nondisabled people, it is necessary to develop a system that can be used not only by teaching staff, but also by the community to help communicate with deaf people. Unlike spoken language, the delivery of sign language is done using body movements as the main means of communication [9]. This shows that sign language can be visualized in the form of images, so that hand movements or facial expressions can be detected automatically through technology. The development of a sign language recognition system aims to make it easier for people to understand sign language [10].

One of the most commonly applied methods for object detection is the convolutional neural network (CNN) method, which is a popular neural network method and is in high demand for object detection [11]. However, another object detection method that can detect objects in real-time with a higher level of accuracy and speed in recognizing objects has been developed, namely the You Only Look Once (YOLO)

method [12]. It was first developed by Joseph Redmon, who proposed a grid-based object detection approach by applying a single convolutional neural network [13]. YOLO is designed as a unified model algorithm that instantly detects and recognizes objects as a whole in a single process [14]. YOLO works by dividing the input image into an S × S grid, where if an object falls into a grid cell, then that grid cell performs the task of detecting the object [15]. The main objective of object detection using YOLO is to find and identify objects in an image from predefined labels by assigning an object class and indicating the position of the object obtained by drawing a bounding box around it [16]. The YOLO method can also detect objects in real-time, making it suitable for systems that require a fast response [17].

Previous case study research applying the YOLO method has been conducted. A research on emotion recognition in facial images using the YOLOv8 method has been conducted [18]. Results indicated that the application of the model successfully detecting emotions through facial images in real-time by processing 400 dataset images consisting of happy, sad, angry and surprised emotions, with a validation rate of mean average precision (mAP) value reaching 90%.

Later, another study was made to detect child abuse and bullying based on YOLOv8 [19]. The research adopted a dataset approach containing the shapes of violent and nonviolent actions with an image size of 640 px. As a result, the model was able to run and detect acts of violence with an accuracy value of 85%, a precision of 81.8%, and a recall of 90%.

Other research has also developed systems to detect the SIBI sign language [20]. The study used a dataset that specialized in translating alphabetical signs from 24 alphabets and was grouped into 4 groups, with each group having 20 image datasets. The results showed that the application of the YOLO method was able to detect gestures with evaluation values from group 1, resulting in an F1 score of 90.90%, group 2 of 97.1%, group 3 of 90.90%, and group 4 of 83.8%.

Some of these studies have applied algorithms from the YOLO model to detect objects and produce good evaluation values. Although object detection research by applying the YOLO method has been carried out, there has been no research study related to sign language identification that is applied in real-time covering datasets from three categories, namely alphabets, numbers and basic words from SIBI, especially data samples obtained directly from the resource person at the State Special School Prof. Dr. Sri Soedewi Masjchun Sofwan, SH. Jambi. This study proposes the development of a sign language recognition system based on computer vision, using a model trained with the You Only Look Once (YOLO) method, which can directly recognize hand movements through a camera without the need for additional auxiliary tools. The developed model is specifically designed for real-time operation and provides recognition results in text and audio formats, enabling two-way communication between deaf and nondeaf people with the aim that the developed model can be an effective communication aid and social interactions between deaf and nondeaf people.

In this study, the model was trained using raw data obtained directly, rather than data sets specifically prepared for local sign language categories. This approach allows for more realistic and practical system development, as the data reflect real-world variations in hand gestures and lighting conditions commonly encountered in daily environments.

## II. METHODOLOGY

This section discusses matters related to the flow of research work, including datasets, data preprocessing, model training, model evaluation, implementation, and system testing.

### A. DATASETS

In this research, the dataset used for the model training process was SIBI. Gesture image data obtained directly from resource person at the State Special School Prof. Dr. Sri Soedewi Masjchun Sofwan, SH. Jambi. The initial stage before taking pictures was the process of interviewing sign experts to determine the class of sign labels that were modeled. The results of the interview showed that the label coverage of the dataset consisted of three categories, namely alphabet, number, and basic words from SIBI. Alphabetical data consists of A–Z, excluding the letter J, which is symbolized as movement. The numeric data consists of *satu–sembilan*, and the basic words consisted of "*bodoh*," "*cinta*," "*jahat*," "*kamu*," "*kasih*," "*maaf*," "*makan*," "*masuk*," "*minum*," "*nama*," "*rumah*," "*saya*," "*terima*," "*tidur*," "*tolong*."

The process of capturing image data from the resource person was carried out using a cellphone camera with a 1:1 aspect ratio and a 12 MP resolution. Sign image data obtained from resource person at the special school were used as reference data for each of the specified label classes. Then, to enrich the content of the dataset, image data were also collected from subjects other than the primary resource person by directly capturing images using cellphone and laptop camera. The laptop camera had a 16:9 aspect ratio and a resolution of 0.9 MP. During the shooting process, the camera was placed at approximately 50–70 cm from the subject, at a height parallel to the subject's chest level. Lighting conditions were maintained using natural daylight or evenly distributed indoor lighting to minimize shadows and ensure clear visibility of hand movements. From this process, 70 primary images were collected for each predetermined label class.

### B. DATA PREPROCESSING

Data preprocessing refers to converting raw data into a format that is easier for machines to understand. Data preprocessing was performed on the Roboflow platform that provides a web-based interface with the stages of image labeling, splitting the dataset, and data augmentation [21]. Image labeling was processed by annotating and grouping data according to the class name of each object in the image to store information related to the image. The image labeling process was processed manually by providing bounding boxes to the image objects one by one to ensure the accuracy of the annotation.

Data that have been annotated and labeled with class information were then be split into three subsets, namely training, validation, and testing subsets: 70% was used as the main material for training the model, 20% was allocated as the validation subset to monitor the performance of the model during the training process, and 10% was used as the testing subset to test the performance of the model that had been trained.

The final stage of data preprocessing was to include augmentation to expand the variety in the data without increasing the amount of primary data. The augmentation methods used were crop, which involved cutting out part of an image so that it no longer had the same position or size; grayscale, which removed color information from images so

that the model could focus more on shapes and patterns; blur, which applied a blurring images to simulate an out-of-focus camera; noise, which added random visual noise to the image, such as small dots or slight distortions; and brightness, which adjusted the brightness of the image to simulate different lighting conditions. The data augmentation process was done by not adding augmentation in the form of image flip because there was the same gesture for two different class labels and was distinguished based on the position of the hand when taking the picture.

### C. MODEL TRAINING

The model training process was conducted using the YOLOv11 model, which was officially released on September 30th, 2024, and served as the main detection framework. The YOLOv11 was selected because it represents the latest development in the YOLO family, offering significant improvements in real-time object detection performance compared to previous versions such as YOLOv8, YOLOv9, and YOLOv10. Among the available YOLOv11 variants (nano, small, medium, large, and x-large), the YOLOv11n (nano) model was chosen for its lightweight architecture, enabling faster inference speeds and real-time implementation on limited graphic processing unit (GPU) resources while maintaining competitive accuracy.

Google Colab was used as the software to operate the model training process. Model training implemented the Tesla T4 GPU runtime to speed up the execution process. Table I shows the combination of hyperparameters used for the success of the training process. The dataset was trained by applying a stepwise training procedure with a maximum number of epochs of 100. The approach aims to obtain the best epoch value that produces a model with optimal performance. If the training process yields a model with an evaluation value in the optimal category, then the training process is stopped even though the epoch value has not reached the maximum value. This strategy is applied to avoid the risk of overfitting, where the model is too explicit on the training data, which results in the model being less able to handle new data because it tends to memorize the training data instead of learning from the training data.

### D. MODEL EVALUATION

An evaluation of the trained model was done by looking at the model's training curve and confusion matrix value. The confusion matrix visualizes the distribution of model prediction errors in a single view. From the basic value of the confusion matrix prediction results, it can be concluded that the evaluation matrix values include precision, recall, accuracy, F1 score, and mean average precision (mAP). The calculation results of the evaluation matrix value can be used to measure the overall performance of the model [22]. The calculation of the evaluation value depends on the base value of the confusion matrix prediction result, which is the main component to get the calculation value of other evaluation matrices, including true positive (TP), true negative (TN), false positive (FP), and false negative (FN). In the precision calculation category, the value is determined from the number of elements that are actually positive, measured against the total positive predicted elements.

$$Precision = \frac{TP}{TP+FP}. \quad (1)$$

TABLE I
HYPERPARAMETER COMBINATION

| No. | Parameters | Value |
|---|---|---|
| 1. | Epoch | Maximum 100 |
| 2. | Batch Size | 16 |
| 3. | IoU | 0.6 |
| 4. | Learning Rate | 0.01 |

Another evaluation matrix is recall, which measures the model's ability to collect all positive elements in a dataset [23]. The recall value is calculated by measuring the ratio of elements that are true positive to the total number of elements that should be predicted as positive.

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

Another evaluation element that becomes an important point to determine the optimization of a model is none other than accuracy, which measures how well an algorithm performs with each data element having the same weight and contributing equally to the Accuracy value.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

From the calculation of the values of the three evaluation matrices above, the F1 score value can also be calculated, which is part of the most frequently applied parametric F measures. F1 score works by calculating the average comparison value of precision and recall [24].

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision+Recall}. \quad (4)$$

The last evaluation matrix value that is considered in this study as a determinant of the optimization of a model is mAP, which is obtained from the average calculated value of average precision (AP). The AP value is obtained from the calculation between precision and recall.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k. \quad (5)$$

By combining all the calculated values of the evaluation matrix, an overall view of the performance of the trained model was obtained. The model with the best evaluation results was used to implement the model into real-time sign language detection.

### E. SYSTEM IMPLEMENTATION AND TESTING

Implementing the model into a real-time system aims to provide users access to interact directly with the model. The training model that had passed the evaluation process and had the best performance was implemented in real-time by utilizing computer vision. The application of computer vision enables computers to "see" and capture objects in the visual environment [25]. The utilization of computer vision in the field of deep learning can be divided into classification, segmentation, detection, and generation, which can provide benefits in various purposes, for example, for security, information, monitoring, and other benefits [26]. The most commonly used task of computer vision is object detection, which refers to the ability of a computer to identify visual objects belonging to certain class, in order to determine whether there are examples of objects of a certain category, such as humans, vehicles, or animals [27]. In this research, the model was implemented using an open-source library commonly applied to systems adopting computer vision,

namely OpenCV. This library provides many task functions, such as motion tracking, object detection, face recognition, and segmentation. By applying OpenCV, real-time images or videos can be adapted according to the needs of the system [28]. The trained model was linked into Python code and OpenCV was employed library to access the webcam camera used as a live image capture tool. The model identification results were shown on the device screen in text form. The identified text was then translated into audio using the Google text-to-speech (gTTS) library. The gTTS libary is a tool that converts text into MP3 format so that it can be saved as an audio file [29]. The generated audio is then played using the Pygame library, specifically the pygame.mixer module, which is used to play audio files [30].

The system that had been successfully run was tested in the final stage to ensure that it could be used in the real environment. In the test, four main aspects of testing were listed: responsiveness, which was to calculate the frame per second (FPS) value of each class label when the system was running using the timing function provided by the OpenCV library to ensure the model could identify objects without significant lag; accuracy, which assessed the level of conformity of the model detection results with the gesture performed by the user; robustness, which evaluated the system's ability to recognize cue objects in various lighting conditions, backgrounds, and other distortions; and voice system, which ensured that the text interpreted into audio was synchronized and running properly.

## III. RESULTS AND DISCUSSION

This section describes the stages that have been carried out in the research to obtain the final results of the research.

### A. DATASET ACQUISITION

Dataset acquisition was done through two procedures in image data capture. In the first procedure, the data sample that became the gesture cue reference for each class label was obtained directly from the resource person at the State Special School Prof. Dr. Sri Soedewi Masjchun Sofwan, SH. Jambi. The data collection process was taken using a cellphone camera, resulting in 49 primary data images corresponding to the number of predefined class labels. Then, the second procedure was carried out to expand the data to a total of 70 primary data images for each class label by adding images taken from subjects outside the resource person while maintaining the gestures from the reference data images. The images in the second procedure were taken using both a cellphone camera and a laptop camera to make the dataset more diverse and representative. The dataset used in this study cannot be made publicly available due to privacy concerns and its large size. However, Figure 1 presents the image data acquisition results as a representative overview of the dataset. The data obtained was stored in one main folder, where each class label was placed in a separate subfolder according to its class label.

### B. DATASET PREPROCESSING

The collected data were prepared for model training by annotating each class label on all images using Roboflow. Each image was annotated individually according to its respective class, and the results were validated directly by the authors in collaboration with the resource person from the State Special School Prof. Dr. Sri Soedewi Masjchun Sofwan, SH. Jambi to ensure labeling accuracy. For the alphabetic and numeric category classes, the labeling was focused on the area of the hand that formed the sign language gesture. Whereas, for the

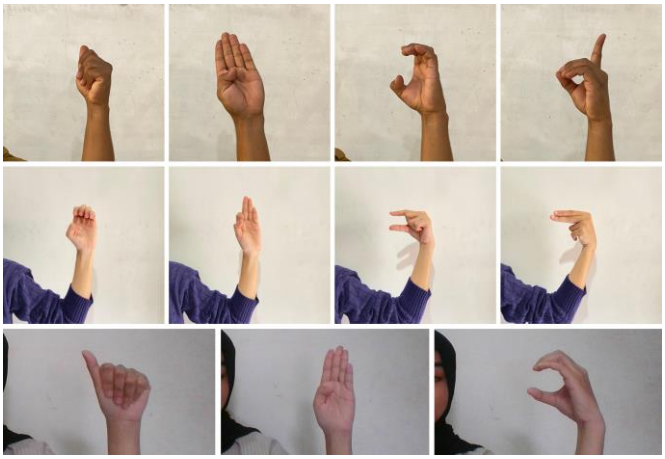

**Figure 1.** Image data acquisition result.

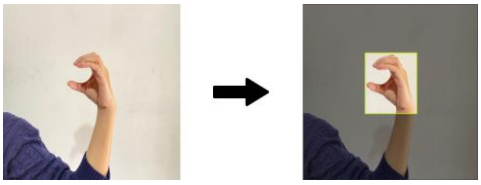

**Figure 2.** Image data annotation.

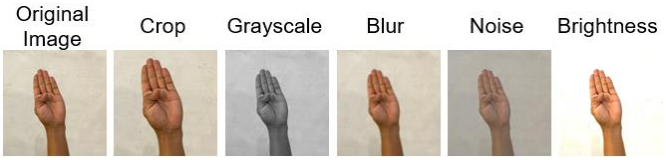


**Figure 3.** Dataset augmentations.

basic words class, the labeling contained the body parts involved in the gesture, such as the face, chest, and arms, depending on the characteristics of each gesture. Figure 2 depict the image labeling process performed on the Roboflow platform.

Once the labeling stage was complete, the annotated dataset was split into three subsets: training, validation, and testing. The total number of primary data images for each class label was 70, so the total number of primary data images obtained in the dataset for 49 label classes was 3,430 images. Of the total number of images, 70% (2,399 images) were allocated to the training subset, 20% (687 images) to the validation subset, and 10% (344 images) to the testing subset.

The final preprocessing step involved augmenting the image to enhance the training subset's dataset. The augmentation process was still done on the Roboflow platform, which could also resize the images to 640 × 640 px so that all images had the same size. Figure 3 depicts the applied augmented image variations in the dataset, including crop, grayscale, blur, noise, and brightness adjustments. The final result of preprocessing produced a total of 5,829 image data in the training subset. This number was obtained because the augmentation process was only applied to the training subset, while the validation and testing subsets were kept without augmentation so that the evaluation results remained objective and represented real conditions.

### C. MODEL TRAINING

The model training process was carried out through incremental epoch cycles, where the entire dataset was processed in each round. In the first training, the model was trained for 10 epochs with a batch size of 16, an intersection

TABLE II
DATASET DISTRIBUTION

| Categories | Number of Classes | Average Initial Images per Class | Total Images | Average Selected Images per Class | Total Images |
|---|---|---|---|---|---|
| Alphabet | 25 | 70 | 1,750 | 40 | 1,000 |
| Number | 9 | 70 | 630 | 40 | 360 |
| Basic Words | 15 | 70 | 1,050 | 40 | 600 |
| Total | 49 | | 3,430 | | 1,960 |

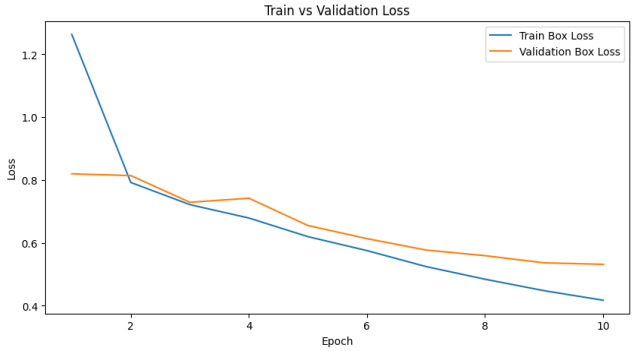


Figure 4. Train vs validation loss curve of the initial model.

over union (IoU) of 0.6, and a learning rate of 0.01. However, the training results obtained did not show good results, since during the training process, the model showed indications of overfitting. This can be seen from the significant difference between the values on the training loss curve, which continues to decrease and reaches a low point. Meanwhile, Figure 4 depicts that the value on the validation loss curve tends to show no significant improvement. The loss value in the train box loss consistently decreased throughout the end of the training process, suggesting that the model could learn and adapt to the parameters used with the training data. However, the loss value in the validation box loss remained stagnant and did not decrease after several epochs. Starting from epoch 4 to epoch 10, the gap between the train box loss and validation box loss curve lines significantly apparent, which indicates the model overfitting.

This situation needs to be improved by performing a model fine-tuning procedure, namely, retraining the pretrained model using the corrected data. Applying the fine-tuning procedure will save more time in the training process than training the model from scratch. The fine-tuning process is carried out by improving the dataset and parameter combinations used during training to obtain more optimal training results.

### *D. FINE–TUNING*

The fine-tuning process was initiated from checking the quality of all the images in the dataset resulting from the preprocessing stage. The findings obtained that were results of data preprocessing captured using a laptop camera decreased in quality from the results of resizing the image. The image looked narrow, and the gesture shape became disproportionate. Therefore, the primary data were reselected by retaining images that were consistent in proportion and object shape for model refinement. From a total of 3,430 primary data images with an average of 70 images per label class, the 40 best quality images were obtained for each label class, bringing the total data used to 1,960. All data were captured using a cellphone camera with a 1:1 ratio. Table II shows the distribution of data before and after the selection process for high-quality images. The selected images then went through the data preprocessing stage again, but with different techniques. The data preprocessing stage to prepare the dataset that was used in the fine-tuning process utilized the library in Python before being re-annotated in Roboflow.

#### 1) FINE–TUNING DATASET PREPROCESSING

The initial stage of the data preprocessing process is to perform the background removal technique on the image. The dataset selected by the fine-tuning process was uploaded into a Google Drive folder and processed using Google Colab. The purpose of removing the background is to remove the background so that the model focuses on the main object in the image without being distracted by other irrelevant elements. The process of removing the background was done by opening all images from the input folder using Python code and executing them using the rembg library. The results of the remove background process were saved using the os.mekedirs function to the remove background results folder.

Dataset preprocessing was followed by a resize process to equalize the image size to 640 × 640 px so that the training process was organized and in accordance with the input image model used. The resize process was done by retrieving image data from the remove background result folder. The resize process was done by utilizing the Python imaging library (PIL), and the image was resized using the thumbnail() method to maintain the aspect ratio of the image. Then, the resized results were saved into a new resized result folder.

To ensure the images used in modeling had good visuals, was not blurry, and still maintained the details of gesture cues, the process of checking image quality using the Laplacian variance method was carried out. The method was used to detect blur based on the calculation of pixel value variations by setting a threshold value of 50, and the passed images were saved into the final folder of preprocessing results using Python.

The last folder used to store the final results of preprocessing using Python was then downloaded, and relabeling, splitting, and augmentation were carried out without entering the resize process in Roboflow. The labeling, splitting, and image augmentation processes followed the same data preprocessing procedure before the fine-tuning process, but were conducted in increments of 10, forming data groups consisting of 10, 20, 30, and 40 primary images data per label class. The final result of data preprocessing was four zipped folders resulting from labeling, splitting, and augmentation according to the amount of primary data.

#### 2) FINE–TUNING MODEL

The model fine-tuning process applied a four-scenario method of gradual data training process, starting from training using 10 primary data images to 40 primary data images per class label. This approach was applied as an effort to anticipate the risk of overfitting, especially considering the limited amount of data. The model with the best performance was used as raw material for system implementation. Table III shows the

TABLE III
MODEL FINE-TUNING TRAINING SCENARIOS

| | Number of Primary Data per Class Label | | | | Number of Data after Augmentation per Class Label | | | | Best Epoch |
|---|---|---|---|---|---|---|---|---|---|
| | *Training* | *Validation* | *Testing* | *Total* | *Training* | *Validation* | *Testing* | *Total* | |
| First Scenario | 7 | 2 | 1 | 10 | 14 | 2 | 1 | 17 | 21 |
| Second Scenario | 14 | 4 | 2 | 20 | 28 | 4 | 2 | 34 | 21 |
| Third Scenario | 22 | 6 | 2 | 30 | 44 | 6 | 2 | 52 | 25 |
| Fourth Scenario | 30 | 8 | 2 | 40 | 60 | 8 | 2 | 70 | 24 |

data distribution for each model fine-tuning scenario and the best epoch obtained. During the fine-tuning model process, the training hyperparameters were readjusted to suit the dataset. After several training trials, the optimal hyperparameter configuration for each fine-tuning scenario was identified. The selected hyperparameters included 40 epochs a patience value of 3, a batch size of 16, and an IoU of 0.6. Additionally, the learning rate was reduced to 0.001, with freeze set to 1 and dropout set to 0.3.

The training process by applying the early stopping mechanism with patience at 3 aimed to stop the training process early if there was no performance improvement during the epoch increments. The freeze and dropout parameters were applied to prevent overfitting by randomly removing some neurons while training the model. During the training process, although the maximum number of epochs was set at 40, in reality, the best results were obtained at no more than 30. For example, for the model in the fourth scenario, the best result is at the 24th epoch. Although the training process continued past that epoch, no significant performance improvement was found, so the 24th epoch was considered the optimal epoch.

### E. MODEL EVALUATION

Model evaluation is done to determine the best model among all completed model training schemes. The model can be categorized as a good model if the evaluation results show optimal model performance and there is no indication of overfitting during the training process. A high evaluation matrix score alone is not enough to conclude that the model is feasible to implement. Models that experienced overfitting were also categorized as unfit because they tended to recognize patterns from the training data and had difficulty in identifying gesture cues when applied in real environments. An indication of the presence or absence of overfitting during the training process can be seen through the train vs validation loss evaluation curve.

Figure 5 depicts the evaluation results of the training process based on the train vs validation loss curve, which indicates that the model in the first scenario did not perform optimally, as the validation loss does not show consistent improvement and diverges from the training loss. The validation loss value on the curve is unstable and often fluctuates up and down during the training process, indicating that the model has not learned optimally from the limited data. However, in the subsequent training process in the second to fourth scenarios of the model, the curve began to show a more stable decreasing pattern suggesting that the error (loss) values on the training and validation data were decreasing, which suggests that the model training process was running well without any indication of overfitting. So, it can be concluded that increasing the amount of data in the training subset can effectively improve the performance of the model.

The performance of each model can also be seen in the confusion matrix value, which gives an idea of the extent to


**Figure 5.** Train vs validation loss curve fine-tuned model.

which the model can predict gesture cues correctly. The confusion matrix is a table that shows the number of model predictions for each label class and compares them with the predicted correct answers (actual labels). The value of the confusion matrix is the main component in calculating the value of other evaluation matrices. Figure 6 depicts the distribution of prediction results for each model fine-tuning scenario across all label classes. The prediction outcomes were then recalculated to obtain additional evaluation metrics, including precision, recall, accuracy, F1 score, and mAP. All figures, including training curves and confusion matrices, were generated directly from Python outputs during the model training process.

Table IV shows the average values of each evaluation component for each model fine-tuning scenario, based on the results of the prediction value calculations. From the evaluation results, the model with the best results was obtained from fine-tuning the fourth scenario model with the training process using 40 primary data for each label class, followed by augmentation to 60 images from each label class, and with the best epoch at 24. The training results showed no indication of overfitting, with the precision evaluation matrix value reaching 99.4%, recall 97.8%, accuracy 99.9%, F1 score 98.7%, and the mAP value 94.3%.

### F. SYSTEM IMPLEMENTATION AND TESTING

The implementation of the model in the real-time system was built using Python program code with the help of Notepad as a text editor. A combination of several libraries needed, such as OpenCV, was used to access the webcam camera, read and display video frames, and display detection results. Then, the gTTS library was used to convert the detected text into audio, and with the help of pygame library, to play the audio directly. The time library is also used to calculate and display the FPS of the detection speed.

The model was run using a local CPU device via Anaconda Prompt as an interface to run the Python environment and the prepared detection scripts that had been prepared. This

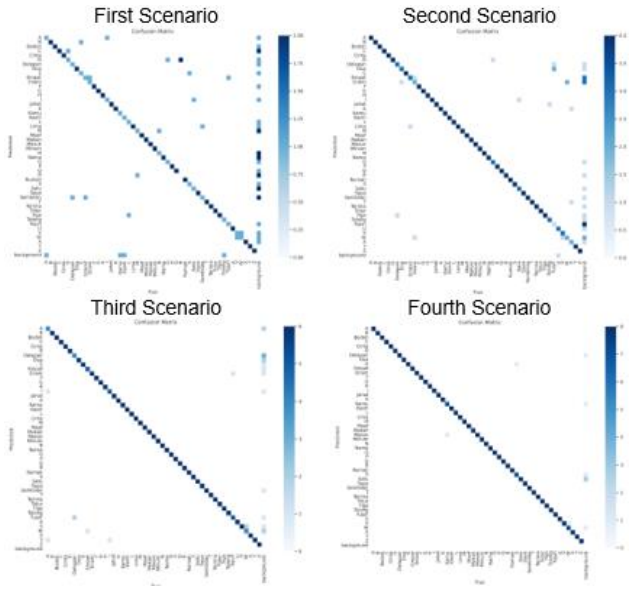


**Figure 6.** Visualization of the confusion matrix prediction fine-tuned model.

TABLE IV
AVERAGE EVALUATION MATRIX VALUE

| | Pretrained Model | | | |
|---|---|---|---|---|
| | ***First Scenario*** | ***Second Scenario*** | ***Third Scenario*** | ***Fourth Scenario*** |
| Precision | 0.770 | 0.929 | 0.975 | 0.994 |
| Recall | 0.695 | 0.935 | 0.956 | 0.978 |
| Accuracy | 0.989 | 0.989 | 0.997 | 0.999 |
| F1 score | 0.816 | 0.904 | 0.948 | 0.987 |
| mAP | 0.911 | 0.894 | 0.928 | 0.943 |

implementation was fully conducted on local hardware without relying on cloud-based resources. Figure 7 depicts the implementation output of the SIBI sign language identification system.

After the system was implemented, testing was conducted to determine whether the system could identify sign language well in the real environment. This test aimed to evaluate the overall performance of the system when run by taking input images from a live camera. The responsiveness aspect was tested by measuring the absence of any significant delay in detecting gesture signals for each label during the detection process. Responsiveness was then calculated based on the FPS value obtained from the average of 10 detection attempts for each class label. The accuracy aspect was evaluated by running the system and recording detection results. Detections were considered accurate if the identified gesture matched the performed gesture and the confidence score > 80%. Robustness was assessed by operating the system under various background and lighting conditions, where successful detection under these conditions was considered a success. Finally, the voice output system was tested to ensure that the generated audio remained synchronized with the recognized text during operation.

After the system was executed and tested, the results showed that the responsiveness of the system in detecting each class label was relatively consistent, with an average of 5 FPS. In this study, no minimum FPS value was defined, as the main focus was to verify whether the system could perform real-time detection without noticeable delay. Although the FPS was relatively low due to the CPU-based implementation, the model still proved effective in recognizing sign language gestures.

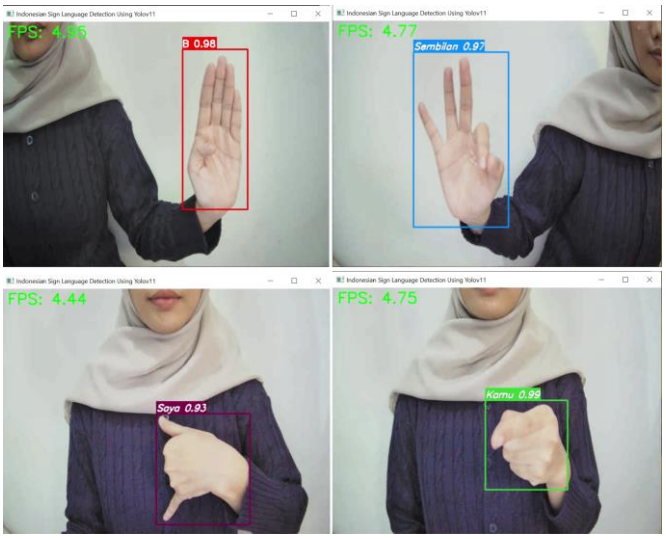


**Figure 7.** Sign language identification system display.

Future improvements in FPS can be achieved by increasing the training data, applying more diverse augmentation, and utilizing more powerful hardware. Overall, the detection results for all class labels were accurate, with confidence scores consistently >80%. However, several class labels, namely "v," "w," "*dua*," and "*enam*" showed lower accuracy due to frequent misclassifications caused by the similarity of their gesture forms. These labels only achieved confidence scores in the range of 50% to 70%. Then, the results of robustness testing showed that the alphabet and number class categories category still had limited detection capabilities because the system was only able to detect these labels in clean background conditions. This is the effect of the dataset used in the model training process, where the dataset has a clean background as the output of the remove background. In the final test, the voice system was evaluated, and the voice output of each label class was always in sync with the text detected by the system.

## IV. CONCLUSION

The results of this research show that sign language, especially SIBI, can be identified in real-time using a YOLO-based training model. The results also show that the data quality, data quantity, and the epoch value significantly affect the quality of the trained model. The YOLO model training in this study achieved the best accuracy during the fine-tuning process in the fourth scenario, with the number of primary data as many as 40 images per label class, with a total of 60 images of subset training after augmentation per label class. The model trained at epoch of 24 produced a training accuracy value of 99.9%, based on the calculation of the confusion matrix evaluation value. Based on the research results, the developed sign language identification system can identify signs through input images taken directly using a webcam and processed in a short time. Thus, the target of developing the YOLO model for identifying sign language has been achieved. However, the model still needs further refinement to address its shortcomings in this study before it can be widely applied in real-world settings. This study will help pave the way for better future research. Future research is recommended to enrich the diversity of backgrounds in the dataset and to adjust the system according to the application domain. Applying more variations of augmentation can possibly overcome the problem of false detection due to similar gestures between labels, and

supplementing the alphabetic dataset by including the alphabet J symbolized in gestures through a video-based data capture and model training approach. For broader applicability, implementing the system in a more accessible form, such as a mobile application or similar platform, is essential for the model to be truly usable in a real-world environment.

## CONFLICTS OF INTEREST

The authors declare that the article entitled "Application of You Only Look Once (YOLO) Method for Sign Language Identification" is written free from conflict of interest.

## AUTHORS' CONTRIBUTIONS

Conceptualization, Reni Triyaningsih, Pradita Eko Prasetyo Utomo and Benedika Ferdian Hutabarat; methodology, Reni Triyaningsih, Pradita Eko Prasetyo Utomo and Benedika Ferdian Hutabarat; software, Reni Triyaningsih; validation, Pradita Eko Prasetyo Utomo and Benedika Ferdian Hutabarat; formal analysis, Pradita Eko Prasetyo Utomo; investigation, Reni Triyaningsih and Pradita Eko Prasetyo Utomo; resources, Pradita Eko Prasetyo Utomo and Benedika Ferdian Hutabarat; data curation, Reni Triyaningsih; writing-original drafting, Reni Triyaningsih; writing-reviewing and editing, Pradita Eko Prasetyo Utomo and Benedika Ferdian Hutabarat; visualization, Reni Triyaningsih; supervision, Pradita Eko Prasetyo Utomo and Benedika Ferdian Hutabarat; project administration, Pradita Eko Prasetyo Utomo and Benedika Ferdian Hutabarat; funding acquisition, Reni Triyaningsih;

## ACKNOWLEDGMENT

The author would like to thank the Faculty of Science and Technology, University of Jambi; and the State Special School Prof. Dr. Sri Soedewi Masjchun Sofwan, SH. Jambi for their invaluable support and assistance in the success of this research. the authors greatly appreciate their cooperation and commitment to advance this research.

## REFERENCES

[1] N.P.L. Wedayanti, A.P. Lintangsari, and G.A.P. Wirawan, "Perkembangan bahasa isyarat daerah Denpasar," *Linguist. Indones.*, vol. 39, no. 2, pp. 217–223, Aug. 2021, doi: 10.26499/li.v39i2.230.

[2] D. Bragg *et al.*, "Sign language recognition, generation, and translation: An interdisciplinary perspective," in *ASSETS '19, Proc. 21st Int. ACM SIGACCESS Conf. Comput. Access.*, 2019, pp. 16–31, doi: 10.1145/3308561.3353774.

[3] M. Sholawati, K. Auliasari, and Fx. Ariwibisono, "Pengembangan aplikasi pengenalan bahasa isyarat abjad SIBI menggunakan metode convolutional neural network (CNN)," *JATI*, vol. 6, no. 1, pp. 134–144, Feb. 2022, doi: 10.36040/jati.v6i1.4507.

[4] A. Pratiwi, "Penggunaan sistem isyarat bahasa Indonesia (SIBI) sebagai media komunikasi (Studi pada siswa tunarungu di SLB Yayasan Bukesra Ulee Kareng, Banda Aceh)," *J. Ilm. Mhs. FISIP (JIMFISIP)*, vol. 4, no. 3, pp. 1–12, Aug. 2019.

[5] I.J. Thira *et al.*, "Pengenalan alfabet sistem isyarat bahasa Indonesia (SIBI) menggunakan convolutional neural network," *J. Algoritma*, vol. 20, no. 2, pp. 421–432, Oct. 2023, doi: 10.33364/algoritma/v.20-2.1480.

[6] R.R.D. Jannah, "Pola komunikasi guru dalam meningkatkan kemampuan belajar siswa tunarungu di sekolah luar biasa negeri Lubuk Linggau," *Wardah*, vol. 22, no. 2, pp. 1–15, Dec. 2021, doi: 10.19109/wardah.v22i2.10830.

[7] E. Juherna, E. Purwanti, Melawati, and Y.S. Utami, "Implementasi pendidikan karakter pada disabilitas anak tunarungu," *J. Gold. Age*, vol. 4, no. 1, pp. 12–19, Jun. 2020, doi: 10.29408/jga.v4i01.1809.

[8] I. Damayanti and S.H. Purnamasari, "Hambatan komunikasi dan stres orangtua siswa tunarungu sekolah dasar," *J. Psikol. Insight*, vol. 3, no. 1, pp. 1–9, Apr. 2019, doi: 10.17509/insight.v3i1.22311.

[9] E. Mustapić and F. Malenica, "The signs of silence – An overview of systems of sign languages and co-speech gestures," *ELOPE, Engl. Lang. Overseas Perspect. Enq.*, vol. 16, no. 1, pp. 123–144, Jun. 2019, doi: 10.4312/elope.16.1.123-144.

[10] Renaldy and A.B. Dharmawan, "Pengenalan citra bahasa isyarat berdasarkan sistem isyarat bahasa Indonesia menggunakan metode vision transformer," *JIKSI (J. Ilmu Komput, Sist. Inf.)*, vol. 12, no. 2, pp. 1–9, Jul. 2024, doi: 10.24912/jiksi.v12i2.31559.

[11] A. Jinan and B.H. Hayadi, "Klasifikasi penyakit tanaman padi mengunakan metode convolutional neural network melalui citra daun (Multilayer perceptron)," *J. Comput. Eng. Sci.*, vol. 1, no. 2, pp. 37–44, Apr. 2022.

[12] Y. Hartiwi, E. Rasywir, Y. Pratama, and P.A. Jusia, "Sistem manajemen absensi dengan fitur pengenalan wajah dan GPS menggunakan YOLO pada platform Android," *J. Media Inform. Budidarma*, vol. 4, no. 4, pp. 1235–1242, Oct. 2020, doi: 10.30865/mib.v4i4.2522.

[13] D.N. Alfarizi *et al.*, "Penggunaan metode YOLO pada deteksi objek: Sebuah tinjauan literatur sistematis," *J. Artif. Intel. Sist. Penunjang Keputusan*, vol. 1, no. 1, pp. 55–63, Jun. 2023.

[14] J.S.W. Hutauruk, T. Matulatan, and N. Hayaty, "Deteksi kendaraan secara real time menggunakan metode YOLO berbasis Android," *J. Sustain., J. Has. Penelit. Ind. Terap.*, vol. 9, no. 1, pp. 8–14, May 2020, doi: 10.31629/sustainable.v9i1.1401.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[16] D. Pestana *et al.*, "A full featured configurable accelerator for object detection with YOLO," *IEEE Access*, vol. 9, pp. 75864–75877, May 2021, doi: 10.1109/ACCESS.2021.3081818.

[17] A. Riansyah and A.H. Mirza, "Pendeteksi mobil berdasarkan merek dan tipe menggunakan algoritma YOLO," *SMATIKA, STIKI Inform. J.*, vol. 13, no. 01, pp. 43–52, Jun. 2023, doi: 10.32664/smatika.v13i01.719.

[18] A. Gallu, A.R. Himamunanto, and H. Budiati, "Pengenalan emosi pada citra wajah menggunakan metode YOLO," *KESATRIA, J. Penerapan Sist. Inf. (Komput. Manaj.)*, vol. 5, no. 3, pp. 1253–1261, Jul. 2024, doi: 10.30645/kesatria.v5i3.444.

[19] G.A. Sidik, "Deteksi tindak kekerasan dan perundungan pada anak berbasis YOLOV8 (You Only Look Once)," *Kohesi, J. Multidisiplin Saintek*, vol. 3, no. 9, pp. 71–80, Jun. 2024, doi: 10.3785/kohesi.v3i9.4044.

[20] B.K. Pratama, S. Lestanti, and Y. Primasari, "Implementasi algoritma you only look once (YOLO) untuk mendeteksi bahasa isyarat SIBI," *J. ProTekInfo*, vol. 11, no. 2, pp. 7–14, Aug. 2024, doi: 10.30656/protekinfo.v11i2.9105.

[21] L. Mahdiyah, S. Oktamuliani, and W.L. Putri, "Penerapan algoritma deep learning YOLOv8 pada platform Roboflow untuk segmentasi citra panoramik," *J. Fis. Unand (JFU)*, vol. 14, no. 3, pp. 228–234, May 2025, doi: 10.25077/jfu.14.3.228-234.2025.

[22] M. Heydarian, T.E. Doyle, and R. Samavi, "MLCM: Multi-label confusion matrix," *IEEE Access*, vol. 10, pp. 19083–19095, Feb. 2022, doi: 10.1109/ACCESS.2022.3151048.

[23] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," 2020, *arXiv:2008.05756*.

[24] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genom.*, vol. 21, pp. 1–13, Jan. 2020, doi: 10.1186/s12864-019-6413-7.

[25] S.V.N. Afni, E.P. Silmina, and I.B. Pangestu, "Computer vision used to monitor the youth during the pandemic COVID-19," in *Procedia Eng. Life Sci.*, 2021, pp. 1–4, doi: 10.21070/pels.v1i2.1019.

[26] T.A. Dompeipen, S.R.U.A. Sompie, and M.E.I. Najoan, "Computer vision implementation for detection and counting the number of humans," *J. Tek. Inform.*, vol. 16, no. 1, pp. 65–76, Mar. 2021, doi: 10.35793/jti.v16i1.31471.

[27] L. Liu *et al.*, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Feb. 2020, doi: 10.1007/s11263-019-01247-4.

[28] H. Adusumalli *et al.*, "Face mask detection using OpenCV," in *2021 3rd Int. Conf. Intell. Commun. Technol. Virtual Mob. Netw. (ICICV)*, 2021, pp. 1304–1309, doi: 10.1109/ICICV50876.2021.9388375.

[29] O.P. Orochi and L.G.Kabari, "Text-to-speech recognition using Google API," *Int. J. Comput. Appl.*, vol. 183, no. 15, pp. 18–20, Jul. 2021, doi: 10.5120/ijca2021921474.

[30] "Pygame," Pygame. Access date: 25-Sep-2025. [Online]. Available: https://www.pygame.org

Jurnal Nasional Teknik Elektro dan Teknologi Informasi
Volume 14 Number 4 November 2025

This page is intentionally left blank