Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.4, August 2026, pp. 3325~3332
e-ISSN: 2548-6861

Real-Time BISINDO Gesture Detection Using YOLOv8 for a Web-Based Text-to-Speech Prototype

Riska Dewi Yuliyanti [1], M. Rafi Muttaqin [2], Teguh Iman Hermanto [3]
* Informatics Engineering, Sekolah Tinggi Teknologi Wastukancana, Purwakarta, Indonesia
riskadewiyuliyanti@gmail.com [1], rafi@stt-wastukancana.ac.id [2], teguhiman@wastukancana.ac.id [3]

Article Info

Article history:

Received 2026-06-16
Revised 2026-07-24
Accepted 2026-08-08

Keyword:

BISINDO,
YOLOv8,
Object Detection,
Gesture Sequence Accumulation,
Text-to-Speech.

ABSTRACT

Communication challenges persist between the deaf and hard-of-hearing community and the general public, largely due to limited awareness and understanding of Indonesian Sign Language (BISINDO) in the broader population. This study develops a real-time web-based system that identifies BISINDO gestures using the YOLOv8 object detection model and converts the recognized gesture sequence into spoken words via a text-to-speech function. The study is based on the CRISP-ML(Q) framework, which includes stages such as understanding the data, preparing the data, building models, assessing their performance, implementing them in real-world applications, and continuously tracking their effectiveness. A total of 1,550 images were independently collected using a laptop camera and categorized into 31 classes, including 30 BISINDO gesture classes and 1 class for negative samples. The dataset was split using a stratified method, allocating 80% for training, 10% for validation, and 10% for testing. The YOLOv8 model was trained on Google Colaboratory using a T4 GPU runtime. The evaluation results indicate that the model achieved a precision of 0.979, a recall of 0.976, an mAP50 of 0.993, and an mAP50-95 of 0.839, which demonstrates robust performance in detecting BISINDO gestures. The developed model was incorporated into a web prototype, with FastAPI serving as the backend and HTML, CSS, and JavaScript utilized for the frontend. The system can identify hand gestures using a webcam, show bounding boxes with corresponding labels, compile the detected gestures into simple text sequences, and produce speech from that information. Latency testing revealed an average response time of around 210 milliseconds when the model was deployed locally and approximately 670 milliseconds when deployed online via Hugging Face Spaces. The system still faces challenges in identifying similar-looking gestures and is influenced by factors such as lighting, hand placement, and the availability of hosting resources. These results indicate that YOLOv8s is effective for detecting the 30 static BISINDO gesture classes evaluated in this study within a controlled data collection setting, though further validation is required before the system can be considered suitable for broader real-world assistive communication use.

I. INTRODUCTION

Communication plays a vital role in how people interact with each other, especially in social settings, learning environments, and public services. People with hearing and speech impairments often face challenges communicating verbally, making it difficult for them to engage with the broader community. Indonesian Sign Language, also known as BISINDO, is one of the main visual communication methods used by the deaf community in Indonesia. However, the lack of widespread knowledge of BISINDO among the general population creates a communication gap that requires technological assistance [1], [2].

Recent advances in computer vision and deep learning have led to growing interest in using image-based methods to recognize sign language. Since sign language includes visual

elements such as hand shape, movement, orientation, and body gestures, computer vision techniques can be used to automatically identify and understand them. Several studies have investigated sign language recognition using methods such as Convolutional Neural Networks (CNNs), MobileNetV2, Long Short-Term Memory (LSTM) paired with MediaPipe, and different versions of the You Only Look Once (YOLO) object detection model [2], [3], [4].

Sign language recognition approaches can generally be grouped into keypoint-based methods, such as MediaPipe Hands combined with classifiers or LSTM networks [5], [6], and object detection-based methods, such as YOLO variants. Keypoint-based approaches provide detailed hand landmark information and have demonstrated high accuracy for gesture recognition in recent studies [5], [6]. However, they typically require an additional preprocessing stage to extract landmarks before classification, and their performance may degrade when hand landmarks are partially occluded or when lighting conditions adversely affect landmark detection. Transformer-based sign language recognition models are capable of capturing temporal and contextual dependencies across longer gesture sequences; however, they generally require larger datasets and greater computational resources, making them less practical for lightweight real-time deployment on resource-constrained web hosting environments. In contrast, object detection-based methods such as YOLOv8 perform localization and classification in a single inference pass directly on raw image frames without requiring a separate landmark extraction step. This characteristic makes YOLOv8 more suitable for the real-time web-based deployment targeted in this study, where inference speed and deployment simplicity on constrained hosting infrastructure (e.g., free-tier CPU-based services) were prioritized over the richer structural representation offered by keypoint- or transformer-based approaches.

YOLO-based models are commonly used for real-time object detection, as they offer a good balance between object identification precision and detection speed. Earlier research has used YOLOv8 for sign language detection and BISINDO alphabet recognition, demonstrating good results in visual gesture detection tasks [1], [7], [8]. Other research has incorporated deep learning models into real-time web-based applications for BISINDO recognition, but many implementations continue to primarily focus on classification or single-stage detection [9], [10].

Recent comparative studies further support the suitability of YOLOv8 for sign language applications. Multiple YOLO versions have been compared for sign language recognition, showing that YOLOv8 achieves superior or competitive performance relative to earlier variants [11]. A systematic review also confirms that YOLOv8 offers a favorable balance between accuracy, inference speed, and computational efficiency across diverse application domains [7].

Within the BISINDO context, strong detection performance has been reported using YOLOv8, achieving an mAP50 of 99.5%, a precision of 95.8%, and a recall of 97.4% [12]. In contrast, a comparative study on YOLOv7 for BISINDO detection reported lower reliability under real-time video conditions [13], further supporting the choice of YOLOv8 for this study.

Even though studies on BISINDO recognition are still advancing, the use of YOLOv8 for gesture detection in a real-time web-based, real-time system with text-to-speech output is uncommon. In real-world applications, a recognition system must not only identify gestures but also display the results in a way that is clear and comprehensible to people who do not use sign language. Therefore, a system that integrates gesture recognition, text collection, and speech generation is necessary to support fundamental assistive communication.

Therefore, this study aims to develop a real-time web-based prototype for detecting BISINDO gestures using the YOLOv8 object detection model. The system is designed to recognize hand gestures using input from a webcam, show the identified gesture along with its corresponding label and a box around the gesture, collect these recognized gestures into a sequence of text, and then turn that text into spoken words using a text-to-speech function. The primary contribution of this study is the integration of YOLOv8-based BISINDO gesture detection into an interactive web prototype, enabling real-time detection and providing speech output as a proof of concept for assistive communication.

Although YOLOv8 has been previously applied to sign language and BISINDO alphabet detection [1], [7], [8], most existing studies focus primarily on isolated gesture or letter classification without addressing how detection results are delivered to end users in a usable form. The novelty of this study lies not in the object detection architecture itself, but in the end-to-end integration of YOLOv8-based BISINDO gesture detection with a real-time web-based pipeline that accumulates detected gestures into a text sequence and converts them into spoken output via text-to-speech. This integration addresses a practical gap identified in prior work [9], [10], where deep learning models for BISINDO recognition were implemented primarily for classification purposes without a complete communication-oriented output pathway.

For instance, an end-to-end system for translating BISINDO gestures into Indonesian text using YOLOv8 combined with a temporal classification refinement framework has been developed, achieving an mAP of 88% [14]. However, the system output remains limited to text and does not extend to speech synthesis. This limitation highlights a gap addressed in the present study, namely the extension of gesture-to-text pipelines into a complete text-to-speech communication output. By combining detection, sequence accumulation, and speech synthesis within a single deployable web prototype, this study contributes a proof-of-concept system architecture for assistive communication, rather than solely a gesture classification benchmark.

II. METHOD

A. Research Framework

This study applies the CRISP-ML(Q) framework, which is a development of the CRISP-DM methodology for machine learning projects. The stages used in this research include data understanding, data preparation, modeling, evaluation, deployment, and monitoring. This framework was selected because the research not only focuses on model training but also includes quality assurance, system deployment, and monitoring of the implemented prototype [15].

The research began with identifying the communication problem between BISINDO users and the general public. The next stage involved collecting and preparing image data for BISINDO gesture detection. After the dataset was annotated and divided into training, validation, and testing subsets, the YOLOv8 model was trained using the prepared dataset. The trained model was then evaluated using object detection metrics, integrated into a real-time web-based prototype, and tested in both local and online deployment environments.

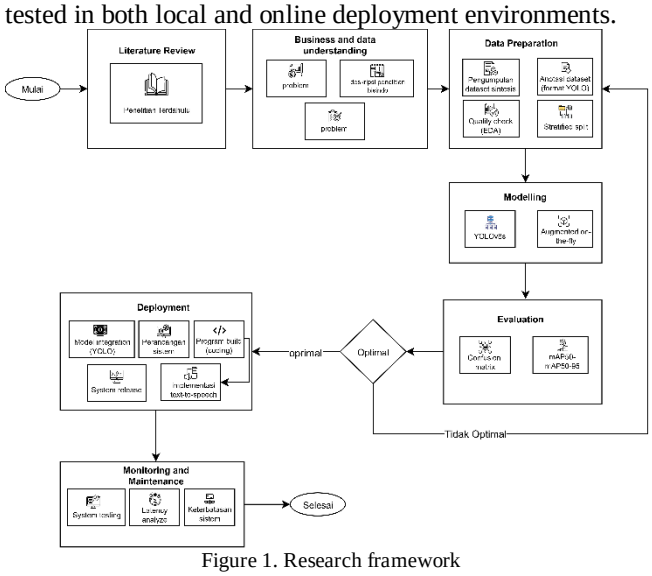

Figure 1. Research framework

B. Dataset Collection and Annotation

The dataset used in this study was collected independently using a laptop camera. The dataset consists of 1,550 images divided into 31 classes. The classes include 30 BISINDO gesture classes and one negative sample class. The gesture classes consist of basic words such as acara, aku, kamu, apa, ayo, bahasa isyarat, bantu, belajar, berdoa, bersama, di jalan, dulu, hati-hati, kapan, kuliah, lelah, maaf, masuk, mulai, nama, pergi, perkenalkan, sebelum, sedang, siapa, sibuk, suka, teman-teman, tolong, and tunggu. The negative sample class contains background images without gesture objects, helping the model distinguish gestures between non-gesture conditions.

The negative sample class consisted of approximately 50 images depicting empty backgrounds, idle hand positions, and other non-gesture frames captured under similar lighting and camera conditions as the gesture classes (Figure 2). This class was included to reduce the likelihood of the model generating false-positive detections when no intentional gesture was being performed in front of the camera, which is particularly important for a continuously running real-time detection system where the camera may capture non-gesture frames between intended signs.

Figure 2. Sample of negative samples

As shown in Figure 2, the negative sample class includes empty backgrounds, idle hand positions, and no-gesture conditions captured under varying lighting conditions. The annotation process was conducted using the Roboflow platform. Each gesture object was labeled with a bounding box corresponding to its class. The annotated dataset was then exported in YOLO format for training the YOLOv8 model. The use of bounding box annotations enables the model to learn not only the gesture class but also its location in the image [16].

C. Data Preparation

After the annotation process, the dataset was prepared using a stratified split method. This method was used to maintain class distribution across the training, validation, and testing subsets. The dataset was divided into 80% training data, 10% validation data, and 10% testing data. The training data consisted of 1,240 images, while the validation and testing data each consisted of 155 images. This data split allows the model to learn from the training data, monitor training performance with the validation data, and evaluate its generalization with the testing data that was not used during training.

TABLE I
DATASET SPLIT

Subset	Images	Percentage
Training	1,240	80%
Validation	155	10%
Testing	155	10%
Total	1,550	100%

D. YOLOv8 Model Training

The model training process was conducted using YOLOv8 in Google Colaboratory with a T4 GPU runtime. YOLOv8 was selected because it is well-suited to real-time object detection and strikes a balance between accuracy and computational efficiency [17], [18]. This study utilized the YOLOv8s (small) variant, chosen to balance detection

accuracy with computational efficiency suitable for real-time web-based deployment. Compared to the larger variants (YOLOv8m/l/x), YOLOv8s offers faster inference speed with a smaller model size, which is advantageous for deployment on limited-resource hosting environments such as Hugging Face Spaces, while still maintaining sufficient accuracy for the gesture classes used in this study.

The training process aimed to produce the best model weights for detecting BISINDO gesture classes. The best-performing model was saved as best.pt and later used for evaluation and system integration.

TABLE II
YOLOv8 TRAINING CONFIGURATION

Parameter	Value
Model architecture	YOLOv8s
Input size	640 x 640
Batch size	16
Maximum epoch	150
Early stopping	Patience = 30
Optimizer	SGD
Initial learning rate	0.01
Final learning rate	0.01
Momentum	0.937
Weight decay	0.001
Label smoothing	0.1

The SGD optimizer was selected based on prior comparative findings for BISINDO detection tasks. A comparative evaluation of SGD, Adam, and AdamW optimizers for YOLOv8-based BISINDO alphabet detection demonstrated that SGD achieved the most consistent performance and the highest mAP@0.5:0.95 (0.942) among the tested optimizers [8]. These results motivated its adoption in this study.

To improve model generalization and reduce overfitting, on-the-fly data augmentation was applied automatically during training. Rather than using the default Ultralytics augmentation configuration, several parameters were adjusted to better reflect real-world variation in gesture capture conditions. Table III summarizes the augmentation configuration used in this study. Rotation augmentation (degrees = 15.0) was applied to simulate natural variation in hand and camera orientation, while scale augmentation (0.2) introduced moderate variation in object size to account for differing distances between the hand and the camera. HSV-based color-space adjustments (hsv_s = 0.4, hsv_v = 0.4) simulated variation in lighting and color saturation across capture sessions. Horizontal flipping (fliplr = 0.5) accounted for left-hand and right-hand gesture variation, mosaic augmentation (mosaic = 1.0) combined multiple training images to expose the model to varied spatial contexts, and copy-paste augmentation (copy_paste = 0.1) was applied to increase object diversity within training samples. These augmentation settings were selected to strengthen the model's robustness against lighting variation, orientation changes, and hand-distance variation without requiring additional manual data collection.

TABLE III
DATA AUGMENTATION CONFIGURATION

Augmentation	Parameter	Value
Scale	Scale	0.2
Rotation	Degrees	15.0
HSV-Saturation	Hsv_s	0.4
HSV-Value (brightness)	Hsv_v	0.4
Horizontal flip	Fliplr	0.5
Mosaic	Mosaic	1.0
Copy-paste	Copy_paste	0.1

E. Web-Based Prototype Development

The trained YOLOv8 model was integrated into a real-time web-based prototype. The system architecture consists of a frontend and a backend. The frontend was developed using HTML, CSS, and JavaScript, while the backend was developed using FastAPI in Python. Communication between the frontend and backend was handled via a REST API.

The system workflow begins when the user activates the webcam through the browser. The captured image frame is sent to the backend through the /predict endpoint. The backend processes the image using the trained YOLOv8 model and returns the detection result in JSON format. The frontend then displays the bounding box, predicted label, and confidence score. The detected gesture label can be accumulated into a simple text sequence and converted into speech using the text-to-speech feature.

The prototype was deployed in two environments: local and online. In local deployment, both frontend and backend were executed on the local machine. For online deployment, the frontend was hosted on Netlify, and the backend on Hugging Face Spaces.

The current system performs gesture recognition on a per-frame basis rather than continuous temporal recognition. Each captured frame is independently classified by the YOLOv8 model, and a detected gesture label is appended to the text sequence only when it is consistently identified above a confidence threshold across a short window of frames, in order to reduce duplicate or spurious detections. This approach differs from continuous/sequential sign language recognition methods that model temporal dependencies across frames (e.g., using LSTM or temporal transformers); rather, it treats each recognized gesture as a discrete unit within an accumulated sequence. This design choice was intended to keep the system lightweight for real-time web deployment, though it does not capture the dynamic, continuous nature of natural sign language production.

F. Evaluation Metrics

The model was evaluated using object detection metrics, including precision, recall, mAP50, and mAP50-95. Precision measures the proportion of correct positive detections among all predicted detections, while recall measures the proportion

of correctly detected objects among all actual objects. The mAP50 metric evaluates detection performance at an Intersection over Union threshold of 0.5, while mAP50-95 evaluates performance across multiple IoU thresholds from 0.5 to 0.95.

In addition to model evaluation, the prototype was tested using black-box testing to ensure that each system function performed its intended purpose. Latency testing was also conducted to compare system response time in local and online deployment environments. Latency testing was conducted using a laptop with an Intel Core i5-8350U CPU @ 1.70GHz (8 CPUs) and 8 GB of RAM for local deployment, and the same device accessing the online deployment through a home Wi-Fi connection with an average download/upload speed of approximately 66.8/39.4 Mbps and a latency of 18 ms. Each latency measurement was repeated 30 times for both local and online environments, and the reported minimum, maximum, and average values in Table V were derived from these repeated measurements to ensure the results were representative rather than based on a single trial.

III. Result and Discussion

A. Dataset Preparation Result

The final dataset used in this study consisted of 1,550 images across 31 classes. Each class was represented in the training, validation, and test subsets via stratified sampling. This distribution was crucial in ensuring that the model learned all gesture classes and could be evaluated more fairly. The inclusion of a negative sample class also helped the model learn non-gesture conditions, reducing the possibility of false detection when no gesture was present in the input frame.

B. Model Training Result

The YOLOv8 model was trained with the configured parameters until it achieved the best validation performance. During training, the model learned to identify the location and class of BISINDO gestures from the annotated images. The use of early stopping helped avoid unnecessary training when the model performance stopped improving. The final best.pt model was used for validation, testing, and integration into the real-time web-based prototype.

C. Model Evaluation

The evaluation results show that the YOLOv8 model achieved strong detection performance on the validation data. The overall precision was 0.979, and the recall was 0.976. The model also achieved an mAP50 score of 0.993 and an mAP50-95 score of 0.839. These results indicate that the model accurately detected BISINDO gestures, especially at the IoU threshold of 0.5.

The high mAP50 value indicates that the model can correctly localize and classify most BISINDO gesture objects. However, the lower mAP50-95 value compared to mAP50 indicates that bounding-box accuracy becomes more challenging under stricter IoU thresholds. This condition may occur because several BISINDO gestures have similar hand shapes, positions, or visual patterns. In addition, variations in lighting, hand distance from the camera, and hand orientation can affect bounding box precision.

Figure 3 presents the confusion matrix generated from the validation results. The matrix shows that the model achieved near-perfect classification across most gesture classes, with the majority of classes exhibiting no cross-class confusion at all. Notably, the three classes with the lowest recall in Table IV, “berdoa”, “kamu”, and “sibuk” were not confused with other gesture classes; instead, one instance from each class was misclassified as background, meaning the gesture was not detected at all rather than being mistaken for a different sign. This indicates that the primary source of error for these classes is missed detection rather than inter-class confusion, which may result from instances where the hand gesture was less distinctly positioned within the frame or affected by momentary motion blur during capture. This finding suggests that improving detection sensitivity for these specific gestures, rather than addressing inter-class visual similarity, would be the more effective direction for future refinement.

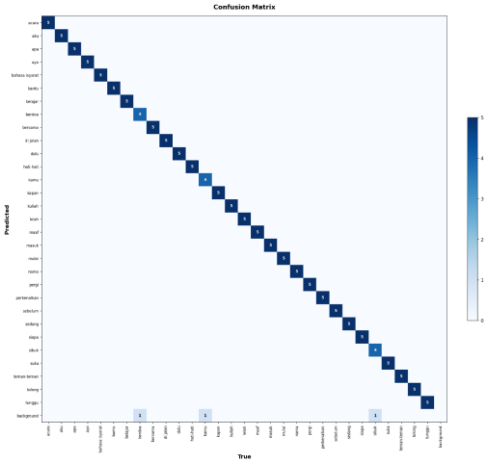

Figure 3. Confusion matrix of YOLOv8 model

TABLE IV
YOLOv8 Model Evaluation

Class	Precision (P)	Recall (R)	mAP50	mAP50-95
all	0.979	0.976	0.993	0.839
acara	0.979	1.000	0.995	0.813
aku	1.000	0.921	0.995	0.812
apa	0.982	1.000	0.995	0.895
ayo	0.975	1.000	0.995	0.875
bahasa isyarat	1.000	0.990	0.995	0.758
bantu	0.979	1.000	0.995	0.881
belajar	0.979	1.000	0.995	0.829
berdoa	0.968	0.800	0.962	0.872
bersama	0.979	1.000	0.995	0.975
di jalan	0.980	1.000	0.995	0.855
dulu	0.977	1.000	0.995	0.905
hati-hati	0.976	1.000	0.995	0.860
kamu	0.971	0.800	0.962	0.805

kapan	0.987	1.000	0.995	0.743
kuliah	0.977	1.000	0.995	0.834
lelah	1.000	0.975	0.995	0.881
maaf	0.975	1.000	0.995	0.907
masuk	0.985	1.000	0.995	0.712
mulai	0.981	1.000	0.995	0.876
nama	0.904	1.000	0.995	0.904
pergi	0.979	1.000	0.995	0.828
perkenalkan	1.000	0.911	0.995	0.855
sebelum	0.961	1.000	0.995	0.868
sedang	0.977	1.000	0.995	0.895
siapa	0.982	1.000	0.995	0.841
sibuk	1.000	0.883	0.995	0.637
suka	0.982	1.000	0.995	0.846
teman-teman	0.978	1.000	0.995	0.761
tolong	0.980	1.000	0.995	0.930
tunggu	0.991	1.000	0.995	0.718

D. Prototype Implementation

The trained YOLOv8 model was successfully integrated into the real-time web-based prototype. The frontend interface allows users to access the camera, send frames to the backend, and view detection results directly in the browser. The backend receives input frames via the /predict endpoint, performs YOLOv8 inference, and returns the predictions to the frontend.

The prototype can display the detected gesture label, bounding box, and confidence score. The detected labels can also be combined into a simple text sequence. Once the text sequence is formed, the text-to-speech feature can convert it into speech. This implementation demonstrates that the proposed system is not only capable of gesture detection but also of supporting simple assistive communication via text and audio output.

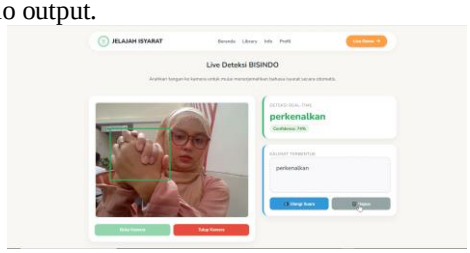

Figure 4. Web-based prototype interface for BISINDO gesture detection

Black-box testing was conducted to evaluate the prototype's main functions. The tested functions included webcam access, gesture detection, bounding box visualization, label display, text accumulation, text-to-speech output, local deployment, and online deployment. The test results showed that all main functions operated successfully and produced the expected output.

Latency testing was conducted in two environments: a local deployment and an online deployment on Hugging Face Spaces. In local deployment, the system achieved a minimum latency of 107 ms, a maximum latency of 582 ms, and an average latency of approximately 210 ms. In online deployment, the system achieved a minimum latency of 430 ms, a maximum latency of 941 ms, and an average latency of approximately 670 ms.

The latency results indicate that local deployment provides faster response time because the backend runs directly on the local device. Online deployment produces higher latency due to network overhead and the limited computational resources of free CPU-based hosting. However, the average online latency remained below 1 second in most cases, indicating that the prototype can still be used in practice for basic real-time interaction.

TABLE V
LATENCY ANALYSIS

Metric	Local	Online
Minimum latency	107 ms	430 ms
Maximum latency	582 ms	941 ms
Average latency	±210 ms	±670 ms

Based on the average inference latency, the corresponding frame processing rate can be estimated at approximately 4.8 frames per second (FPS) in local deployment (1000 ms / 210 ms) and 1.5 FPS in online deployment (1000 ms / 670 ms). While these rates are lower than typical continuous video frame rates (e.g., 24–30 FPS), it is important to note that the system does not require continuous video-rate processing, as gesture recognition is performed on a per-request basis triggered by discrete captured frames rather than a continuous video stream. Nonetheless, the relatively low FPS in online deployment indicates that further optimization, such as model quantization, reduced input resolution, or GPU-based hosting, would be necessary to support smoother real-time interaction at scale.

F. System Limitations

Although the proposed system achieved strong evaluation results, several limitations remain. First, the detection performance is affected by lighting conditions, hand position, camera quality, and distance between the hand and the camera. Although systematic testing with quantitative metrics under varied lighting, background, and distance conditions has not yet been conducted, informal observation during system demonstration indicated that the model remained capable of detecting gestures reliably under slightly suboptimal lighting conditions, although performance declined when the camera distance from the user was too far. A similar sensitivity to environmental conditions such as dynamic lighting and complex backgrounds has also been reported in related real-time sign language detection research [19]. This preliminary finding motivates our recommendation to enrich the variation in camera distance and lighting conditions during dataset collection in future work. Second, several BISINDO gestures share similar visual characteristics, which may lead to misclassification or reduced bounding-box accuracy. Third, the dataset used in this study remains limited in both the number of images per class and the variation across users, backgrounds, and environments.

Another limitation concerns deployment performance. The online version of the system uses free CPU-based hosting,

which affects response time and system stability. In addition, the gesture sequence accumulation method is still simple because it relies on the order of detected gestures over time and does not yet include a complete linguistic analysis of BISINDO grammar. It should also be noted that this system is intentionally scoped to static hand gesture detection as the initial phase of this research, and does not yet cover non-manual BISINDO elements such as facial expressions and dynamic body movements, which also play an important role in the meaning and grammar of sign languages as a natural language. This scoping decision is consistent with related YOLOv8-based sign language detection studies that similarly focus on manual components and identify non-manual component recognition as a direction for future development [20]. The system has also not been validated by BISINDO language experts, so the output sequence should be considered a basic proof of concept rather than a complete translation system.

In addition to the limitations described above, several further constraints should be acknowledged. First, the 30 gesture classes used in this study represent only a small subset of the broader BISINDO vocabulary, which consists of a considerably larger number of signs used in everyday communication; the current system should therefore be regarded as a proof-of-concept covering basic conversational vocabulary rather than a comprehensive BISINDO recognition system. Second, the dataset was collected from a limited number of individuals using a single laptop camera, and therefore does not capture the full range of natural variation in hand size, skin tone, left- versus right-hand dominance, and individual signing style that would be encountered among the broader BISINDO-signing population. This limits the extent to which the reported evaluation metrics can be generalized to unseen users. Third, misclassification was observed to concentrate on gesture classes with visually similar hand shapes or finger configurations, such as *berdoa*, *kamu*, and *sibuk* (Table IV), suggesting that fine-grained hand-shape discrimination remains a challenge for frame-based object detection approaches; addressing this may require higher-resolution hand-region cropping, additional discriminative features, or hybrid approaches that incorporate keypoint information alongside object detection. Finally, evaluation of the text-to-speech output quality and overall communication effectiveness has not yet been assessed through direct testing with BISINDO users or members of the deaf community; the current evaluation remains limited to technical model performance and system-level functional testing, and future work should include usability studies involving the target user population to validate real-world communication effectiveness.

IV. CONCLUSION

This study developed a real-time web-based prototype for BISINDO gesture detection using the YOLOv8 object detection model and a text-to-speech feature. The dataset consisted of 1,550 images across 31 classes, including 30 BISINDO gesture classes and 1 negative sample class. The dataset was split using a stratified method into 80% training data, 10% validation data, and 10% testing data.

The YOLOv8 model achieved strong detection performance, with a precision of 0.979, a recall of 0.976, an mAP50 of 0.993, and an mAP50-95 of 0.839. These results indicate that YOLOv8s can effectively detect the specific set of static BISINDO gestures evaluated in this study, though this finding reflects technical feasibility under controlled data collection conditions rather than a generalized conclusion about YOLOv8's suitability for BISINDO recognition as a whole or for deployment as a complete communication system. The trained model was successfully integrated into a web prototype using FastAPI as the backend and HTML, CSS, and JavaScript as the frontend. The system can detect gestures through a webcam, display bounding boxes and labels, accumulate detected gestures into simple text, and convert the text into speech.

Latency testing showed that the system achieved an average response time of approximately 210 ms in local deployment and 670 ms in online deployment. The difference was mainly due to network overhead and limited computational resources during online deployment. Although the prototype can be used in practice as a proof of concept, several limitations remain, including sensitivity to lighting and hand position, limited variation in the dataset, a simple accumulation of gesture sequences, and the absence of expert validation by BISINDO practitioners.

Beyond its technical contribution, the proposed prototype demonstrates potential practical value for supporting basic communication between BISINDO users and the general public in everyday settings, such as simple public service counters, classroom introductions, or informal social interactions where a hearing person may not understand sign language. However, realizing this potential in practice would require further collaboration with the deaf community and BISINDO practitioners to validate vocabulary relevance, ensure cultural and linguistic appropriateness, and adapt the system to more complex real-world usage scenarios such as healthcare or educational settings.

Future research should expand the dataset by involving more users, backgrounds, lighting conditions, and gesture variations. Further development should also improve temporal gesture recognition, involve BISINDO experts for validation, optimize deployment performance, and explore more efficient models for real-time web or mobile implementation.

ACKNOWLEDGMENT

The author would like to express gratitude to Sekolah Tinggi Teknologi Wastukancana Purwakarta, the Informatics Engineering Study Program, and the supervising lecturers for their guidance and support during this research.

BIBLIOGRAPHY

[1] M. M. Fresmanda, Istiadi, and S. W. Irianandа, "Deteksi Objek Video Bahasa Isyarat Untuk Anak Tuna Rungu dan Tuna Wicara Menggunakan YOLOv8," *Jurnal Komputer, Informasi dan Teknologi*, vol. 4, no. 2, Oct. 2024, doi: 10.53697/jkomitek.v4i2.1895.

[2] F. I. Hadinata and S. A. Sanjaya, "BISINDO Sign Language Recognition: A Systematic Literature Review of Deep Learning Techniques for Image Processing," *Indonesian Journal of Computer Science Attribution*, vol. 12, no. 6, p. 3281, Dec. 2023.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2021. [Online]. Available: http://pjreddie.com/yolo/

[4] H. Y. A. Swasono, A. R. Himamunanto, and F. Maedjaja, "Implementasi YOLO11 dan OpenCV untuk Pengenalan Frasa dalam Video Real-Time Bahasa Isyarat Tangan," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 5, no. 3, Jul. 2025, doi: 10.57152/malcom.v5i3.2130.

[5] A. Ahmad Ilham and I. Nurtanio, "Dynamic Sign Language Recognition Using Mediapipe Library and Modified LSTM Method," vol. 13, no. 6, 2023.

[6] B. Subramanian, B. Olimov, S. M. Naik, S. Kim, K. H. Park, and J. Kim, "An integrated mediapipe-optimized GRU model for Indian sign language recognition," *Sci. Rep.*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-15998-7.

[7] N. A. Megantara and E. Utami, "Sistemasi: Jurnal Sistem Informasi Object Detection Using YOLOv8 : A Systematic Review," 2025. [Online]. Available: http://sistemasi.ftik.unisi.ac.id

[8] Y. T. Buana and D. Ariatmanto, "Comparative Evaluation of Optimizer in YOLOv8 for BISINDO Alphabet Detection," 2025. [Online]. Available: http://publishing-widyagama.ac.id/ejournal-v2/index.php/jointecs

[9] E. L. Kelana, M. R. A. Prasetya, Mambang, and M. Zulfadhilah, "Integrating the CNN Model with the Web for Indonesian Sign Language (BISINDO) Recognition," 2025. [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAIC

[10] N. Thalbiatul *et al.*, "Pengembangan Aplikasi Web Pengenalan Huruf Bahasa Isyarat Indonesia (BISINDO) Real-Time Menggunakan Mobilenetv2," vol. 10, no. 2, 2025.

[11] Mangai V and Kalaimagal R, "Comparative Analysis of Various Yolo Models for Sign Language Recognition with a specific dataset," 2024. [Online]. Available: https://www.jisem-journal.com/

[12] A. B. Pangestu, R. Muttaqin, and A. Sunandar, "Sistem Deteksi Bahasa Isyarat Indonesia (BISINDO) Menggunakan Algoritma You Only Look Once (YOLO)V8," 2024.

[13] N. Renaningtias, F. Putra Utama, A. Nur, and A. Sobri, "Deteksi Bahasa Isyarat Indonesia (BISINDO) Pada Video dengan YOLOv7," *JSAI: Journal Scientific and Applied Informatics*, vol. 8, no. 1, 2025, doi: 10.36085.

[14] S. Putra and E. Rakun, "End-to-end system for translating bahasa isyarat Indonesia sign language gestures into Indonesian text," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 40, no. 2, p. 719, Nov. 2025, doi: 10.11591/ijeecs.v40.i2.pp719-734.

[15] S. Studer *et al.*, "Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology," *Mach. Learn. Knowl. Extr.*, vol. 3, no. 2, pp. 392–413, Jun. 2021, doi: 10.3390/make3020020.

[16] J. Murrugarra-Llerena and C. R. Jung, "Can we trust bounding box annotations for object detection?," 2022. [Online]. Available: http://www.cvlibs.net/datasets/kitti/

[17] J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Dec. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/make5040083.

[18] M. Hussain, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection," Jul. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/machines11070677.

[19] M. Hasan, B. K. Paul, N. Islam, and R. Mostafiz, "Advancing real-time sign language detection for deaf and hearing-impaired communities: a customized YOLOv8 approach with tailored annotations in computer vision," *BMC Artificial Intelligence*, vol. 1, no. 1, Oct. 2025, doi: 10.1186/s44398-025-00010-9.

[20] 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, Jan. 2024, doi: 10.3233/JIFS-235132.