Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.3, June 2025, pp. 883~896
e-ISSN: 2548-6861

Integrating the CNN Model with the Web for Indonesian Sign Language (BISINDO) Recognition

Enisda Libra Kelana [1]*, Muhammad Riko Anshori Prasetya [2]*, Mambang [3]*, Muhammad Zulfadhilah [4]*
* Informatics, Universitas Sari Mulia
enisdalibra23@gmail.com[1], riko.anshori@gmail.com[2], mambang@unism.ac.id[3] , zulfadhilah@unism.ac.id[4]

Article Info

Article history:

Received 2025-03-26
Revised 2025-06-14
Accepted 2025-06-17

Keyword:

Deep Learning,
Indonesian Sign Language,
Sign Language Recognition,
Web.

ABSTRACT

Effective communication is challenging for deaf individuals in Indonesia, most of whom use Indonesian Sign Language (BISINDO). Sign Language Recognition (SLR) can bridge this communication gap. While Convolutional Neural Networks (CNNs) show high potential for SLR, their practical accessibility remains limited. This research aims to develop a CNN architecture for recognizing BISINDO alphabet signs from static images (still images) and integrate it into an accessible web platform. Using a static vision-based approach, a CNN model was trained on a public dataset (312 images, 26 classes) following standard pre-processing including data augmentation. The model was subsequently integrated into a web interface using Python and the Gradio library. Results demonstrated strong model performance, with validation accuracy reaching 97.44% and a macro-average F1-score of approximately 97.12%. However, classification challenges were identified for visually similar signs ('M' and 'N'). The resulting integrated web application proved functional, exhibited low prediction latency, and showed cross-platform compatibility. This study successfully demonstrates the development of an accurate DL model for static BISINDO alphabet recognition and its practical implementation via a web platform. This contributes to reducing the accessibility gap in SLR technology. Future research is recommended to utilize larger, more varied datasets and explore dynamic sign recognition.

I. INTRODUCTION

Communication is an important aspect of social life. Through communication individuals can interact with the surrounding environment, convey ideas, and obtain education [1]. However, there are individuals who have limitations in communicating verbally due to hearing or speech impairment they are deaf - speech impaired [2], [3]. The Population Census conducted in 2020 showed that 1.43% of Indonesia's population were people with disabilities, with 0.36% having hearing impairments and 0.35% having speech impairments [4].

Sign language is a language used by people with deaf-speech disabilities to communicate. In Indonesia there are two sign languages: The Indonesian Sign Language System (SIBI) and Indonesian Sign Language (BISINDO). SIBI is a standardized system developed by the Indonesian government and adapted from American Sign Language (ASL) [5]. In contrast, BISINDO emerged naturally and serves as the primary language for the majority of Deaf individuals in Indonesia [6]. Research shows that only 9% of people with disabilities use SIBI and 91% use BISINDO [7].

Sign Language Recognition (SLR) is a method that aims to facilitate communication between deaf and hearing individuals. The field of research focuses on the automatic identification of signs within specific sign languages and their subsequent translation into formats such as text or speech, thereby rendering the signed communication accessible to non-signers [8], [9].

Sign Language Recognition generally uses two approaches, which are vision-based approach and sensor-based approach. In the sensor-based approach, sensor devices are attached to the body to capture the position and movement of hands, fingers and other body parts. These sensors will

generate signals that are then processed for Sign Language recognition [8], [9], [10]. This sensor-based approach has the disadvantage that sensor devices are expensive [10]. Meanwhile, the vision-based approach takes hand gesture images as input and converts them into text or voice. This vision-based approach can utilize the Traditional Machine Learning approach and Deep Learning (DL) approach. Traditional approaches use Hidden Markov Models (HMM) and Support Vector Machines (SVM), while non-traditional approaches utilize Deep Learning such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) [7], [8], [10], [11], [12]. Deep Learning is a subfield of Artificial Intelligence that focuses on learning the representation of data by utilizing artificial neural networks inspired by the structure and function of the human brain, DL has many layers of processing [13].

Vision-based approaches to sign language recognition can be classified into static and dynamic methods, depending on the type of visual input analyzed [9]. Static approaches emphasize the extraction of pertinent spatial characteristics from static images of isolated signs, such as alphabets, and generally utilize two-dimensional convolutional neural network (CNN) architectures. Conversely, dynamic approaches process video sequences to simultaneously capture spatial and temporal information, including motion and transitions between signs [13]. This capability is essential for continuous sign language recognition (CSLR) tasks. Consequently, dynamic methods often necessitate more complex architectures and tend to require greater computational resources [13].

This research employs a vision-based sign language recognition approach to identify static signs corresponding to the BISINDO alphabet. The selection of this methodology was driven by its accessibility and its independence from costly, specialized hardware or substantial computational resources. The implementation of this vision-based recognition offers significant practical advantages, as it can be readily utilized with standard camera devices integrated into smartphones or laptops [9].

Vision-based approaches employing Deep Learning (DL) techniques, such as Convolutional Neural Networks (CNN), have demonstrated promising accuracy in recognizing Indonesian Sign Language [7], [11], [14], [15], [16], [17]. However, the real-world application of these models remains significantly limited, hindering their broader accessibility and use [8], [10]. Therefore, the objective of this research is to address this limitation by integrating a developed DL model into a web-based sign language recognition platform, specifically designed with an emphasis on practicality and user-friendliness.

II. METHOD

A. Research Pipeline

This study employs a quantitative approach to identify static hand gestures associated with Indonesian Sign Language (BISINDO), utilizing a deep learning (DL) model and integrating it into a web-based interface. The research was conducted by following the flow in Figure 1, which began with conducting a literature study of related research articles both national and international articles. After conducting a literature study of various sources, the researcher continued by determining the research topic, determining this topic after carefully reading the existing literature. At this stage the researcher determines the topic of Sign Language Recognition. The problem formulation stage is done by asking questions related to what solutions can be done to the problems encountered in the topic being studied. Data collection is a stage where researchers collect related data, this involves finding appropriate data, downloading, and pre-processing before being given to the model for the training stage. The next stage is model building and model evaluation. The choice of DL architecture to be used is closely related to the type of data used. This research will use CNN because of its ability to recognize patterns in images by maintaining the relationship between pixels in the image. Models that have been trained using data will be evaluated to ensure the prediction results provided by quality models. Web design stages include system design, web interface design. Integrating the model with the web is the stage of implementing the model with the web. The last stage is report writing, at this stage all experimental results are written systematically. The flow of research conducted can be seen in Figure 1.

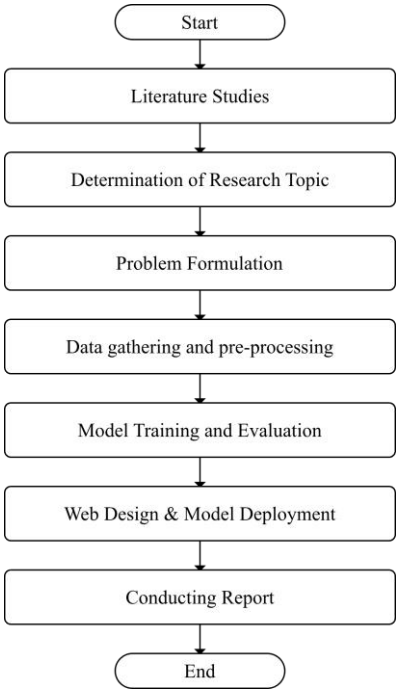

Figure 1. Research Process

As shown in Figure 1, this research was conducted step by step in a sequential manner from literature review to report writing. The process is depicted from top to bottom.

B. Data Collection

This study utilizes secondary data, specifically comprising open-source images of hand gestures representing the Indonesian Sign Language (BISINDO). The data selection criteria applied in this research are detailed in Table 1.

TABLE I
DATA SELECTION CRITERIA

Criterion	Specification
Data Type	Open-source image data
Data Source	Kaggle or UCI Dataset
Data Format	Images (JPG, PNG)
Sign Language System	BISINDO (Indonesian Sign Language)
Minimum Resolution	200 x 200 pixels

The data search process will be modified to align with the data criteria specified in Table 1. This search will be conducted within the Kaggle and UCI open-source data repositories. The process of data search is shown in Figure 2.

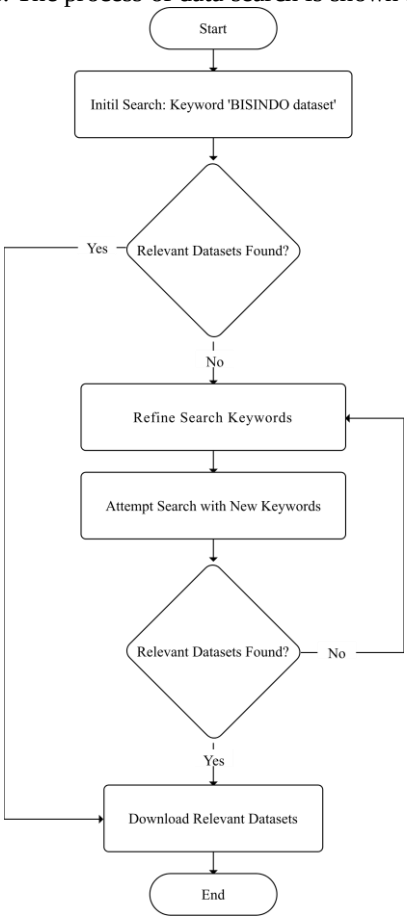

Figure 2. Data Collection Process

The data collection process, as illustrated in Figure 2, included search initiation, keyword refinement, and dataset downloading.

1) *The initiation of the search process:* Querying open-source data websites using the keyword "BISINDO dataset". The Kaggle and UCI databases were selected due to their provision of straightforward access to a wide array of relevant and contemporaneous open-source datasets.
2) *The refinement of the keywords:* This is a necessary step when the initial keywords are not matching any relevant datasets.
3) *Dataset downloading:* Matched datasets are then downloaded for further processing.

During the searching process, one dataset was found that was relevant to the required data criteria. This data was obtained from the Kaggle repository under the title Indonesian Sign Language (BISINDO) Alphabets, created by Achmad Noer. The dataset contains a total of 312 images, and consists of 26 classes corresponding to the alphabet (A-Z), with each class containing 12 BISINDO alphabet images [18].

The sample BISINDO alphabet dataset can be seen in Figure 3. In Figure the hand gestures of the letters A, B, C, D, E and F in BISINDO are shown.

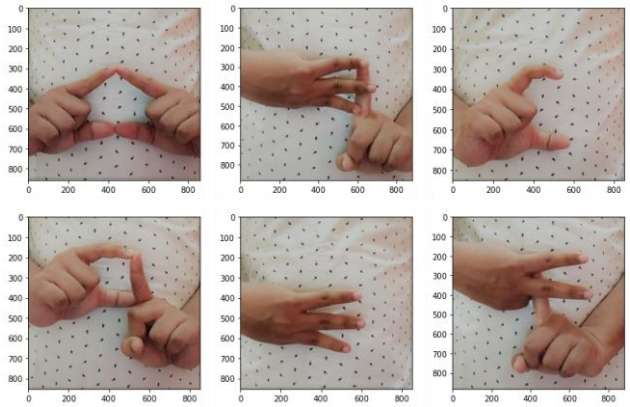

Figure 3. Sample images from the Indonesian Sign Language dataset for the letters A, B, C, D, E, and F.

The comprehensive information regarding the Indonesian Sign Language (BISINDO) Alphabets Dataset is shown in Table 2.

TABLE II
DATASET INFORMATION

Feature	Description
Data Type	Images
Format	JPEG (.jpg)
Colour Space	RGB
Dimensions (pixels)	512×512
Number of Classes	26
Label	A, B, C, ..., Z
Images per Class	12
Total Number of Images	312
Is there a class imbalance?	No

C. Pre-processing

The downloaded data will be adjusted to the appropriate size and format before being input into the model. This stage is termed pre-processing [10]. This pre-processing stage involves several key steps, including the splitting of the data into training and validation sets, encoding, and augmentation.

1) Data Splitting: As illustrated in Figure 4, modifications to the directory structure occur during the data splitting phase of pre-processing. The left panel, designated as "Before Data Splitting," illustrates the initial structure of the dataset, where all image data resides in a single data directory, categorized into classes (schematically represented as directories A through Z). The right panel ("After Data Splitting") illustrates the outcome of the aforementioned partitioning. The original data directory is divided into two distinct subsets: a training set (train) and a validation set (val). Notably, the original class structure (A through Z) is maintained within both the training and validation sets, making sure that each subset contains examples from all classes. The dataset utilized in this study comprises 312 images, equally distributed across 26 distinct sign language classes, resulting in 12 samples per class.

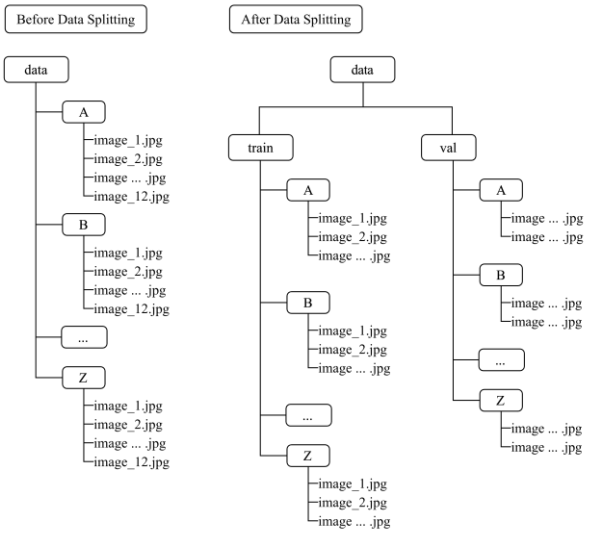

Figure 4. Data Splitting

Standard percentage-based data splitting methods (e.g., 80% training, 20% validation) were deemed potentially suboptimal due to the limited number of samples per class [19]. Such methods could lead to significant class imbalance or even zero representation for some classes in the validation set. To mitigate this issue and ensure robust evaluation across all categories, a specific data partitioning strategy was adopted. Instead of a conventional percentage ratio, a fixed number of samples—specifically, 3 images (designated as support) from each of the 26 classes—were allocated to constitute the validation set. This allocation resulted in a validation set containing 78 images (3 samples/class × 26 classes) and a training set comprising the remaining 234 images, effectively yielding a 75% training and 25% validation split ratio. This method guarantees balanced class representation within the validation set, thereby providing a more reliable basis for evaluating the model's generalization performance across all classes.

2) Label Encoding: Label encoding was performed because deep learning models generally operate more effectively with numerical data compared to categorical inputs. Figure 5 illustrates this label encoding process, a prerequisite step in preparing the dataset for deep learning. The dataset, located within the main data directory, is organized into a training set ('train') and a validation set ('val'). Categorical class labels, initially represented by folder names corresponding to letters ("A", "B", "C" through "Z"), were systematically mapped to numerical identifiers. Consequently, as depicted, each class is now represented by a numerically labelled subdirectory (ranging from 01 to 26) within both the 'train' and 'val' folders.

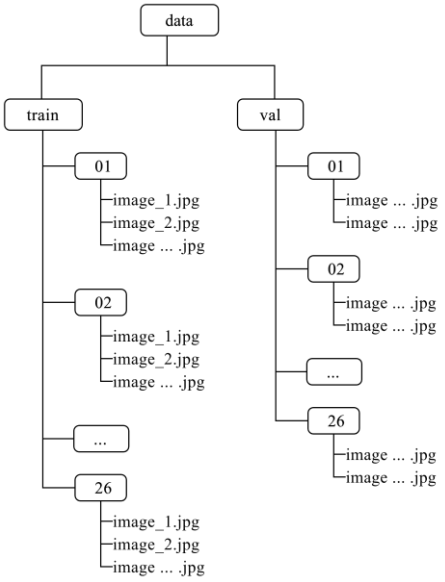

Figure 5. Label Encoding

This numerical representation is crucial for enabling the deep learning model to effectively process and learn from the class distinctions during the training and evaluation phases.

3) Data Augmentation: Data augmentation was implemented to address the challenge of limited data availability by synthetically expanding the dataset from existing images. Data augmentation is a widely employed technique in deep learning and computer vision, leveraged to enhance model performance and generalization capabilities [20]. This approach involves increasing the diversity of the training dataset by applying various transformations, such as image rotation, to the existing data. The specific augmentation techniques and parameters utilized in this study are detailed in Table 3 Augmentation Parameters.

TABLE III
AUGMENTATION PARAMETERS

Parameter	Value	Description
rescale	1./255	Rescales pixel intensity values from the [0, 255] range to the [0, 1] range by multiplying with the specified factor (1/255).
rotation	10	Randomly rotates images by an angle selected uniformly from the range [-10, +10] degrees.

width_shift	0.2	Randomly shifts images horizontally by a fraction of the total width, selected uniformly from the range [-0.2 * width, +0.2 * width].
heigh_shift	0.1	Randomly shifts images vertically by a fraction of the total height, selected uniformly from the range [-0.1 * height, +0.1 * height].
zoom_range	[0.8, 1.2]	Randomly applies zoom to the image by selecting a zoom factor uniformly from the specified range. A factor < 1 zooms in, > 1 zooms out. E.g., `[0.8, 1.2]` corresponds to zooming in/out by up to 20%.
brightness_range	[0.8, 1.2]	Randomly adjusts image brightness by picking a brightness shift factor uniformly from the specified range `[lower, upper]`. E.g., `[0.8, 1.2]` modifies brightness between 80% and 120%.
color_shift_range	10.0	Randomly shifts color channel values (e.g., R, G, B) by adding an intensity value selected uniformly from the range [-10.0, +10.0].

Table 3 illustrates the application of data augmentation to a sample BISINDO representing the letter 'A'. This process involves a series of geometric and photometric transformations designed to artificially enrich the training dataset, thereby enhancing the robustness and generalization capability of the classification model when faced with input variations

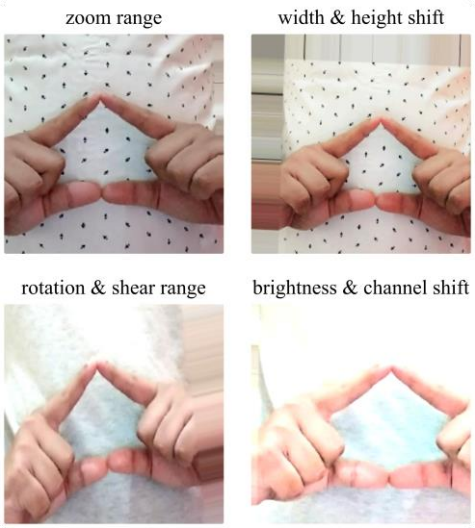

Figure 6. Results of the image augmentation process

As illustrated in Figure 6, the applied transformations encompass a diverse set of modifications. Specifically, the augmentation techniques employed include zoom, width shift, height shift, rotation, shear, brightness shift, and channel shift. The specific parameters that govern the degree and range of each transformation, which yielded the augmented data samples displayed, are defined by the values specified in Table 3. The selection of these augmentation parameters and value ranges was not arbitrary, but was determined through a series of empirical trials. These trials aimed to identify the optimal configuration for effectively simulating real-world variability without introducing excessive distortion to the original gesture data.

D. Deep Learning Model Building

The development process for the Deep Learning (DL) model encompasses several key stages: architecture definition, model training, performance evaluation, and subsequent saving of the trained model. Architecture definition constitutes the initial and fundamental phase. During this phase, the network configuration is specified, including the number of layers, the quantity of units within each layer, and the filter sizes to be employed, particularly for the feature extraction task from image data. Table 4 provides detailed specifications of the layers and their corresponding parameters utilized in this study for this image feature extraction purpose.

TABEL IV
MODEL PARAMETERS

Layer	**Output Shape**	**Param**
conv2d	(None, 254, 254, 16)	448
max_pooling2d	(None, 127, 127, 16)	0
conv2d_1	(None, 125, 125, 32)	4640
max_pooling2d_1	(None, 62, 62, 32)	0
conv2d_2	(None, 60, 60, 64)	18496
max_pooling2d_2	(None, 30, 30, 64)	0
conv2d_3	(None, 28, 28, 128)	73856
max_pooling2d_3	(None, 14, 14, 128)	0
conv2d_4	(None, 12, 12, 256)	295168
max_pooling2d_4	(None, 6, 6, 256)	0
conv2d_5	(None, 2, 2, 512)	3277312
max_pooling2d_5	(None, 1, 1, 512)	0
flatten	(None, 512)	0
dropout	(None, 512)	0
dense	(None, 512)	262656
dense_1	(None, 256)	131328
dense_1	(None, 26)	6,682

After the architecture is defined, the model compilation step is performed. This stage involves the selection of the loss function, the optimization algorithm (optimizer), and the evaluation metrics to be monitored during the ensuing training phase. The specific parameters employed for model compilation and training are detailed in Table V.

TABLE V
TRAINING PARAMETERS

Parameter	**Value**
Input Shape	256*256
Batch Size	64
Optimizer	RMSprop
Learning Rate	0.001 (default)
Loss	Categorical Crossentropy
Epoch	200

During the training stage, the model learns to map patterns in hand gesture images to their corresponding alphabet labels within the dataset. This training was conducted using Google Colab, leveraging T4 GPU acceleration.

Subsequent to the training stage, the model undergoes a stage of evaluation in which its performance is assessed. In this phase, the trained model is evaluated on previously unseen hand gesture images (i.e., the validation set) to ascertain its capacity to generalize and accurately recognize signs.

Finally, the model saving stage involves storing the evaluated model for subsequent integration into the sign language recognition web application. This process involves the storage of the model's architecture, compilation parameters, and learned weights in the HDF5 (.h5) file format.

E. BISINDO recognition web design

The web design stage includes system design which aims to describe how user interactions with the model through the web and web interface design.

1) Use case diagram: The description of the interaction between the user and the DL model through the web is described through the UML use case diagram and can be seen in Figure 7 system Design Using Use case Diagram.

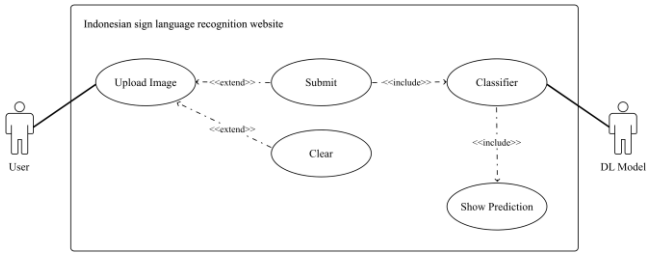

Figure 7. System Design Using Use Case Diagram

2) User Interface Design: After the design of the interaction between the user and the model through the web is finalized, proceed to the next stage, namely the design of the Indonesian Sign Language Recognition web interface as can be seen in Figure 8.

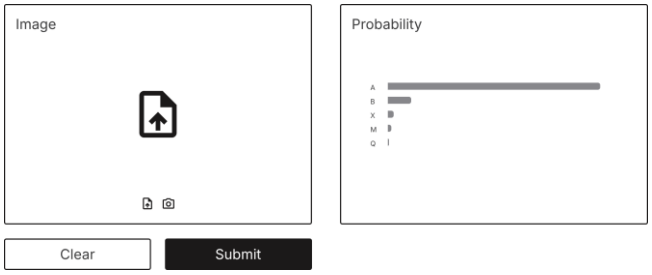

Figure 8. Design of BISINDO Recognition Web Page

3) Functional Description of User Interface Components: The user interface (UI) is used as the primary medium through which users interact with the Indonesian Sign Language (BISINDO) recognition web application. To ensure effective navigation of the system and utilization of its sign recognition capabilities, it is imperative that users possess a comprehensive understanding of the function of each button, display area, and input element. Consequently, this section provides functional details for these UI elements. Table 6 below systematically presents comprehensive information on each web interface component, outlining its specific function and purpose within the application workflow.

Following the finalization of the web interface design, the study proceeded to the crucial subsequent phase: the integration of the previously developed deep learning model into the web environment. This integration was technically implemented using the Python programming language. Specifically, the Gradio library (package) was utilized in this implementation to facilitate the creation of an interactive interface, bridging the model's functionality with the end-user via the web application [21].

III. RESULTS AND DISCUSSION

A. Deep Learning Model

The training of the deep learning model was conducted for a total of 200 epochs. Upon completion of this training process, the model attained a final accuracy of 94.02% on the training dataset and demonstrated strong generalization performance with an accuracy of 97.44% on the validation dataset. The progression of model accuracy throughout the training phase, illustrating the learning convergence, is depicted graphically in Figure 9.

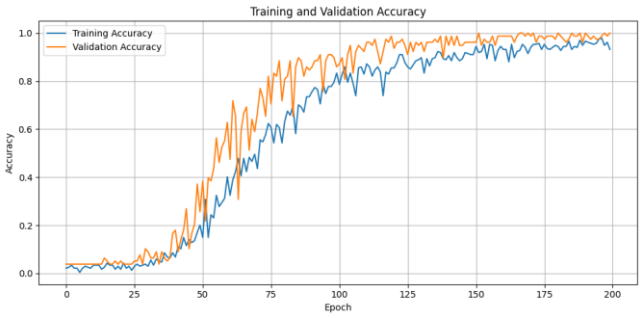

Figure 9. Training and Validation Accuracy

Figure 9 shows the accuracy of the model during the training period in the first 50 epochs, the training accuracy of the model increases slowly, identifying progress in the model learning process. Meanwhile, in the early epochs there is a significant accuracy recall in the validation accuracy, indicating the effective generalization ability of the model. In the range of 50 to 100 epochs, the training and validation

accuracy increased sharply with the training accuracy showing a steady increase, while the validation accuracy fluctuated. In the final epoch phase, the accuracy diagram is more stable indicating that the model has achieved consistent performance. The subsequent sections will provide a detailed evaluation of the model's performance using standard evaluation metrics.

1) Classification Report: The detailed performance evaluation of the proposed classification model across all 26 classes is summarized in the classification report presented in Table 7. This report provides key performance metrics, including precision, recall, F1-score, and support, offering insights into the model's effectiveness on a per-class basis.

TABLE VII
CLASSIFICATION REPORT

class	precision	recall	f1-score	support
A	1.00	1.00	1.00	3
B	1.00	1.00	1.00	3
C	1.00	1.00	1.00	3
D	1.00	1.00	1.00	3
E	1.00	1.00	1.00	3
F	1.00	1.00	1.00	3
G	1.00	1.00	1.00	3
H	1.00	1.00	1.00	3
I	1.00	1.00	1.00	3
J	1.00	1.00	1.00	3
K	1.00	1.00	1.00	3
L	1.00	1.00	1.00	3
M	0.60	1.00	0.75	3
N	1.00	0.33	0.50	3
O	1.00	1.00	1.00	3
P	1.00	1.00	1.00	3
Q	1.00	1.00	1.00	3
R	1.00	1.00	1.00	3
S	1.00	1.00	1.00	3
T	1.00	1.00	1.00	3
U	1.00	1.00	1.00	3
W	1.00	1.00	1.00	3
X	1.00	1.00	1.00	3
Y	1.00	1.00	1.00	3
Z	1.00	1.00	1.00	3

The evaluation results for the Indonesian Sign Language (BISINDO) recognition model across 26 distinct classes are detailed in Table 7. The majority of classes, specifically 01-12 (A-L) and 15-26 (O-Z), achieved perfect scores (1.00) for precision, recall, and F1-score. This performance indicates highly effective classification for these categories within the evaluation dataset. However, notable exceptions were observed: Class 13 (M) demonstrated lower precision (0.60) despite perfect recall (1.00), resulting in an F1-score of 0.75. Conversely, Class 14 (N) achieved perfect precision (1.00) but suffered from low recall (0.33), yielding a correspondingly lower F1-score of 0.50.

Confusion Matrix: The performance evaluation of the proposed deep learning model for recognizing 26 Indonesian Sign Language (BISINDO) hand signs is presented via a confusion matrix, as illustrated in Figure 10. This matrix provides a visual representation of the model's classification performance on the test dataset, with its predictions mapped against the actual class labels. The findings reveal that the model attained a high degree of accuracy across a wide range of sign classes. The superior performance of the model is evident from the predominance of values along the main diagonal of the matrix, with all test samples (n = 3 per class) for 25 out of the 26 sign classes being correctly classified. However, a misclassification between sign classes '14' and '13' was identified. Specifically, two of the three samples classified as '14' were misclassified as '13'. Consequently, the model accurately identified only one sample from class '14.' No other misclassifications were observed among the remaining sign classes during this evaluation.

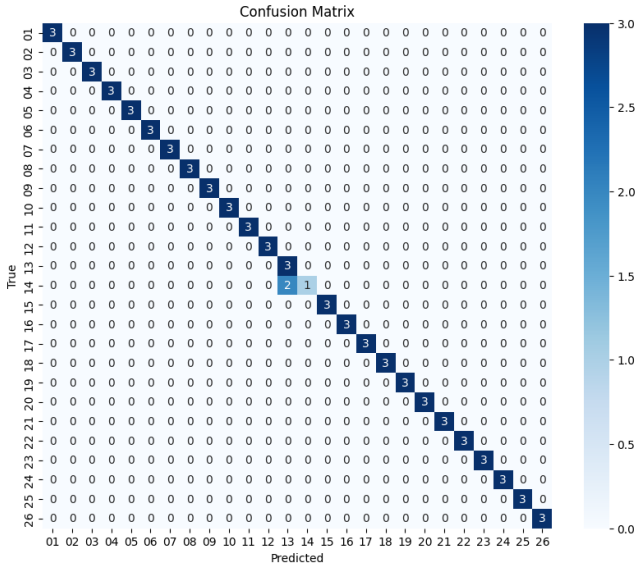

Figure 10. Model Evaluation Using Confusion Matrix

The findings demonstrate the model's robust discriminative capability, despite the challenges in differentiating the visual characteristics between signs '13' and '14.' This suggests the potential for visual similarity between these two sign classes.

2) Overall Performance Metrics: This classification process specifically involves N=26 distinct classes (categories), featuring a balanced sample distribution across classes, with each class uniformly represented by 3 support samples. Consequently, the entire evaluation dataset comprises a total of 78 samples, derived directly from the multiplication of the total number of classes by the number of support samples per class (26 classes × 3 samples/class). This class-balanced structure constitutes an important characteristic of the evaluation set employed in this study.

Class-Specific Metric Derivation:

For classes 01 to 12 and 15 to 26, the following equations are to be used:

- TP = 3
- FN = 0

- FP = 0
- TN = Total Sample - TP - FP - FN = 78 - 3 - 0 - 0 = 75

For Class 13, the values are as follows:
- TP = 3
- FN = 0
- FP = 2
- TN = Total Sample - TP - FP - FN = 78 - 3 - 2 - 0 = 73

For Class 14, the values are as follows:
- TP = 1
- FN = 2
- FP = 0
- TN = Total Sample - TP - FP - FN = 78 - 1 - 0 - 2 = 75

Macro Averaging: Macro-averaging is an approach used to calculate aggregate performance metrics, such as precision or recall, in multi-class classification tasks. This approach entails the preliminary calculation of the target metric for each class independently. The simple arithmetic mean of these individual per-class scores is then computed to yield the final aggregate score. A fundamental implication of this method is that it assigns equal weight to each class in the computation of the final average score, irrespective of the actual number of samples (support) within each class.

Macro-averaged Precision: This macro-averaging approach assesses model performance by weighting each class equally. It is computed by averaging the precision scores calculated independently for each class. The general formula for Macro Precision is:

$$\text{Macro Precision} = \frac{1}{\text{l}}\sum_{\text{i=1}}^{\text{l}} \frac{\text{tp}_\text{i}}{\text{tp}_\text{i} + \text{fp}_\text{i}}$$

...where l is the total number of classes (26 in this study), tpi represents the true positives, and fpi denotes the false positives for the i-th class. The Macro Precision calculation involves summing the precision values from all 26 classes (yielding a total of 25.60), which is then divided by the number of classes (l). The result of the calculation, 2625.60, is approximately 0.9846. This value represents the model's average precision performance across all classes, assigning equal weight to each class regardless of its sample size.

Macro Average Recall: This metric is a measure of the model's average ability to independently identify all true positive instances for each class. It is computed by taking the arithmetic mean of the recall scores calculated individually for each class. The formula for Macro Recall is as follows:

$$\text{Macro Recall} = \frac{1}{\text{l}}\sum_{\text{i=1}}^{\text{l}} \text{Recall}i = \frac{1}{l}\sum \text{i} = 1^{\text{l}} \frac{\text{tp}_\text{i}}{\text{tp}_\text{i} + \text{fn}_\text{i}}$$

...where l is the total number of classes, tpi represents the true positives, and fni denotes the false negatives for the i-th class. Macro Recall calculation was performed by first summing the recall scores from all classes. Based on the per-class results, 24 classes exhibited perfect recall (1.00), another class also had a recall of 1.00, while the one remaining class had a recall of 31. The total sum of these recall scores is $(24 \times 1.00) + 1.00 + 31 = 376$. This total value was then divided by the number of classes (26) to obtain the macro-average: $2676/3 = 7876$. The final result of this Macro Recall calculation is approximately 0.9744, indicating that the model, on average, demonstrates a very high capability to recognize positive samples from each class.

Macro F1-Score: The F1-Score represents the harmonic mean of precision and recall, providing a single metric that balances these two performance aspects, a characteristic particularly valuable in the presence of class imbalance. Consistent with other macro-averages, this method assigns equal weight to each class, irrespective of its sample size. The formula employed for the Macro F1-Score is:

$$\text{Macro F1} = \frac{1}{\text{l}}\sum_{\text{i=1}}^{\text{l}} \text{F1}_\text{i}$$

...where l is the total number of classes and F1i is the F1-Score for the $\text{i} - \text{th}$ class, calculated as

$$\text{F1}_\text{i} = \frac{2 \times \text{Precision}_\text{i} \times \text{Recall}_\text{i}}{\text{Precision}_\text{i} + \text{Recall}_\text{i}}$$

In its implementation in this research with 26 classes (l=26), the Macro F1-Score calculation involved summing the F1-Scores from all classes. Based on the per-class evaluation, it was found that 24 classes achieved a perfect F1-Score (1.00), one class had an F1-Score of 0.75, and another class had an F1-Score of 0.50. The total sum of these F1-Scores is $(24 \times 1.00) + 0.75 + 0.50 = 25.25$. The macro-average value was then calculated by dividing this total sum by the number of classes (26), yielding $\frac{25.25}{26}$. The final result of the Macro F1-Score calculation is approximately $0.9712 \approx 97{,}12\%$, which indicates excellent overall model performance in balancing precision and recall evenly across all classes.

Weighted Averaging: Weighted averaging calculates the mean of per-class metrics, weighting each class's score by its support. This method accounts for class imbalance, contrasting with the macro-averaging approach where all classes are weighted equally. However, given the uniform support of 3 samples per class in this study's validation set, the weighted average calculation is mathematically equivalent to the macro average, consequently yielding the same numerical result.

Accuracy: The primary evaluation metric computed is Accuracy. This is the most intuitive measure of classification model performance, representing the proportion of total samples across all classes that the model classifies correctly. This metric offers a general overview of the model's overall correctness in making predictions. The sum of these values is calculated across all classification categories. The formula employed for the Accuracy is:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

In the formula above, TP represents true positives, TN represents true negatives, FP represents false positives and FN represents false negatives. In the context of the model evaluation in this research, specifically on the validation dataset comprising 78 samples (with a balanced representation of 3 samples per class), the model was recorded as making 76 correct predictions. Thus, its accuracy calculation is:

$$\text{Accuracy} = \frac{76}{78} = 0.97435897 \ldots \approx 97.44\%$$

This high accuracy value indicates that the model demonstrates excellent overall predictive correctness on the dataset used for this evaluation.

Performance Analysis of Classes 13 and 14: Based on the evaluation metrics presented above, this section provides a detailed performance analysis for Class 13 (letter M) and Class 14 (letter N). The recognition model demonstrated low precision for Class 13 (letter M), achieving a score of 0.60. This indicates that when the model predicted a sign as M, the prediction was correct only 60% of the time. The remaining 40% comprised signs from other classes that were misclassified as M, signifying a high False Positive rate for this class. Conversely, the model achieved perfect recall (1.00) for Class 13, successfully identifying all actual instances of the M sign within the evaluation dataset. Consequently, no M signs were missed (zero False Negatives). This suggests a model tendency to over-predict Class 13; while capturing all true M signs, it also erroneously classifies other signs as M. Specifically, the confusion matrix Figure 10 reveals that two samples belonging to Class 14 (letter N) were misclassified as M. In contrast, Class 14 (letter N) exhibited perfect precision (1.00). This implies that every instance predicted as N by the model was indeed correct, resulting in no False Positives. However, the model's recall for this class was notably low at 0.33 (approximately 1/3).

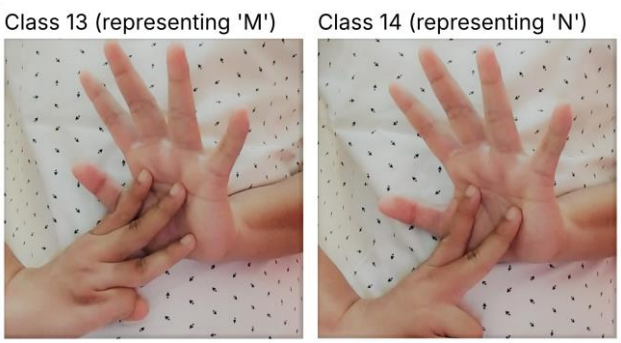

Figure 11. Visual similarity between Class 13 ('M') and Class 14 ('N')

This signifies that the model only correctly identified 33% of the actual N signs present in the evaluation data. A significant majority (67%) of true N signs were consequently misclassified as other letters, indicating a high False Negative rate. This pattern suggests the model adopts a highly conservative approach when predicting the N sign, leading to many true instances being overlooked. This classification difficulty may be attributed to the visual similarity between the signs for M and N, as potentially illustrated in Figure 11.

As illustrated in Figure 11, representative images are presented for sign class M (depicted on the left) and sign class N (depicted on the right). A high similarity between these two sign classes can be observed. The distinguishing factor between these two sign classes lies in the number of fingers that make contact with the palm: three fingers for class M, and two fingers for class N.

B. Model Integration

After evaluating the DL Model, the next step is to integrate the model with the web. Implementation of the model with the web is done using the Python Programming Language. Figure 12 shows the initial appearance of the BISINDO recognition web.

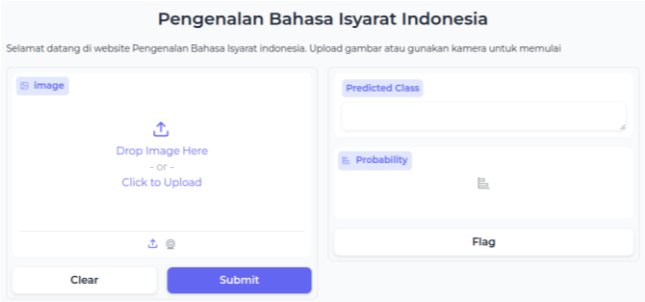

Figure 12. Initial View of BISINDO Recognition Web

Figure 13 shows the BISINDO recognition web interface when hand gesture image recognition is run. On the left side, the image preview will appear and on the right side, the class prediction results will appear.

Figure 13. Web View when Sign Language Recognition is Running

Figure 13 shows that the integration between the model and the BISINDO recognition web has been successful, this is shown by the web integrated with the model that successfully recognizes the letter “F” in the BISINDO hand gesture in a short time.

C. Further Evaluation

Robustness to Lighting Conditions: In this study, the performance evaluation of the static hand sign recognition model was conducted under three distinct lighting conditions: normal, dim, and dark. Ideally, image data acquisition would be performed directly under each specific lighting condition to accurately represent real-world environmental variations. However, due to resource and time constraints encountered during the research period, direct data capture under dim and dark conditions was not feasible. As a pragmatic alternative approach, these lower light conditions were simulated through digital manipulation of the images originally captured under normal lighting. This simulation involved applying a uniform black color overlay (hex code #000000) to the original images, setting the opacity level to 60% to represent dim conditions and 80% to represent dark conditions. The corresponding results, demonstrating the model's robustness to these lighting conditions, are presented in Table 8.

TABLE VIII
ROBUSTNESS TO LIGHTING CONDITIONS

Sign	Normal	Dim	Dark	Notes
A	84	85	53	None
B	84	76	98	None
C	88	71	24[1]	[1]Misclassified as letter D
D	94	74	39[2]	[2]Misclassified as letter B
E	90	85	86	None
F	78	87	29[3]	[3]Misclassified as letter G
G	72	56	39	None
H	92	89	74	None
I	94	82	86	None
J	92	76	57	None
K	94	63	47	None
L	96	64	40	None
M	89	76	58	None
N	57	41	37[4]	[4]Misclassified as letter M
O	91	88	92	None
P	90	47	27[5]	[5]Misclassified as letter V
Q	88	52	45	None
R	85	68	45	None
S	96	77	65	None
T	80	61	69	None
U	80	70	70	None
V	85	91	78	None
W	89	82	85	None
X	82	85	48	None
Y	78	82	64	None
Z	79	55	24[6]	[6]Misclassified as letter V

While acknowledging the inherent limitations of this simulation method compared to using natively captured data under varied lighting, this approach enabled a preliminary investigation into the model's robustness against reduced light intensity within the scope of the existing constraints. The presented table compares the confidence scores (probabilities) assigned by the model to the actual class label for static hand signs (A-Z) when evaluated under three distinct lighting conditions. Under normal lighting conditions, prediction confidence was generally high (majority >80%), although some inter-class variation was observed. Dim lighting led to a heterogeneous decrease in confidence across classes. This decrease became more drastic and widespread under very dark conditions, with the majority of classes exhibiting scores below 50%, indicating high prediction uncertainty. Nevertheless, under both reduced lighting conditions, inter-class performance variability was evident, with some classes demonstrating greater robustness than others

1) Prediction Latency Analysis: Table 9 shows the results of the computation time required by the model to predict a single hand sign (latency). These results were obtained under three different lighting conditions: Normal, Dim, and Very Dark. The latency tests were performed for each sign language class from A to Z.

TABLE IX
PREDICTION LATENCY ANALYSIS

Isyarat	Normal	Dim	Dark
A	0.066	0.099	0.099
B	0.066	0.102	0.075
C	0.071	0.071	0.067
D	0.098	0.069	0.068
E	0.067	0.067	0.065
F	0.066	0.067	0.068
G	0.065	0.094	0.065
H	0.074	0.064	0.067
I	0.064	0.066	0.065
J	0.072	0.096	0.064
K	0.144	0.064	0.066
L	0.066	0.065	0.085
M	0.065	0.066	0.064
N	0.065	0.064	0.087
O	0.070	0.096	0.081
P	0.144	0.067	0.068
Q	0.066	0.101	0.066
R	0.144	0.144	0.067
S	0.111	0.101	0.073
T	0.073	0.083	0.075
U	0.086	0.072	0.100
V	0.066	0.091	0.065
W	0.065	0.065	0.064
X	0.066	0.096	0.065
Y	0.064	0.109	0.078
Z	0.064	0.113	0.078

According to the data in Table 9, the model's prediction time (latency), measured in seconds, was generally very fast across all lighting conditions, with most predictions completed in under 0.1 seconds per sign. Further analysis indicated no consistent trend suggesting that darker lighting conditions (both dim and very dark) significantly and uniformly affected the model's prediction speed. The impact of lighting changes on latency appeared variable across sign classes, with certain signs exhibiting minor, non-systematic decelerations or accelerations under different lighting conditions. Furthermore, some instances of relatively higher latency were observed under normal conditions for specific signs (e.g., K, P, R).

TABLE VI
FUNCTIONAL DESCRIPTION OF USER INTERFACE COMPONENTS

Component ID	Type	Precondition(s)	Function	Interaction & Response
upload	Icon Button	None	Initiates the selection and loading of a BISINDO hand gesture image from the user's local storage.	Opens the operating system's file selection dialog. Upon selection, the image is displayed in the *Image* preview area. Enables the `submit` and `clear` buttons.
camera	Icon Button	Device has a camera; User grants camera access permission (if required).	Activates the device's camera interface for capture BISINDO sign image.	Activates the device's camera stream. The captured image is displayed in the `image_preview_area`. Enables the `submit` and `clear` buttons.
image_preview	Image Preview Area	Image loaded via `upload` or captured via `camera`.	Displays the hand gesture image selected or captured by the user, prior to submission for classification.	Displays image data received from `upload` or `camera` actions. Content is cleared by the `clear` action.
submit	Primary Button	A valid image is displayed in the Image preview area.	Initiates processing of the displayed image data using the integrated classification model.	Passes image data to the integrated model. Receives classification results (predicted class, probability score) directly from the integrated model and updates the corresponding output areas.
clear	Secondary Button	An image is displayed in the Image preview area.	Resets the image input area, removing the currently displayed image.	Removes the image from the *Image* preview area. Disables the `submit` button until a new image is provided (via `upload` or `camera`).
`predicted_class`	Text Output Area	Successful completion of integrated model processing following `submit`.	Displays the predicted class label for the submitted hand gesture image, as determined by the model.	Content is populated by the system based on the results returned by the local model function/process. Displays the resulting class name (e.g., "A", "B", "Hello").
`probability`	Text Output Area	Successful completion of integrated model processing following `submit`.	Displays the confidence score (probability) associated with the predicted class provided by the model.	Content is populated by the system based on the results returned by the local model function/process. Displays the probability score (e.g., "98.5%", "0.985").

2) Web Functional Testing: To verify that the sign recognition web application functions as expected across various user environments, functionality testing was conducted. Table 10 summarizes the results of this functionality testing, focusing on key application components across different combinations of web browsers and operating systems. The testing procedure was conducted online and involved 10 university student participants. These participants were asked to access the web application from various devices and then complete an online research questionnaire that included usability testing questions.

TABEL X
PREDICTION LATENCY ANALYSIS

Browser (version)	Operating Systems	Component ID						
		upload	camera	preview	submit	clear	predicted_class	probability
Chrome (135.0.7049.85)	Windows 10	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Firefox (137.0.1)	Windows 10	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Chrome (135.0.7049.84)	Arch Linux	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Firefox (137.0.1)	Arch Linux	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Chrome (135.0.7)	Android 13	Passed	Passed	Passed	Passed	Passed	Passed	Passed
Firefox (137.0)	Android 13	Passed	Passed	Passed	Passed	Passed	Passed	Passed

The functionality testing results, presented in Table 10, indicate that all key components of the web application—including image upload, camera access, image preview, submission for prediction, input clearing, as well as the display of results (predicted_class and probability) - successfully passed testing ('Passed') across all tested configurations. The tested configurations encompassed the Chrome and Firefox web browsers on the Windows 10, Arch Linux, and Android 13 operating systems. These results indicate that the application's core functionality performs correctly and exhibits sufficient cross-browser and cross-platform compatibility across the tested environments.

D. Further Evaluation

1) Dataset Comparison: To provide context for the dataset utilized in this study, a comparison was made with datasets employed in previous related research focusing on Indonesian Sign Language (BISINDO). Table 11 presents this comparison, detailing key characteristics of these datasets, including the number of images, acquisition source, number of classes, and their corresponding reference citations.

TABEL XI
COMPARISON OF DATASET CHARACTERISTICS USED IN RELATED STUDIES

Reference Study	Number Images	Acquisition Source	Number Classes	Notes
BISINDO (Bahasa Isyarat Indonesia) Sign Language Recognition Using CNN and LSTM [11]	1.100	Author's Collection (Private)	10	None
Indonesian Sign Language Recognition using YOLO Method [22]	4.547	Author's Collection (Private)	24	None
Indonesia Sign Language Recognition using Convolutional Neural Network [14]	39.455	Author's Collection (Private)	37	None
Convolutional Neural Network (CNN) for Image Classification of Indonesia Sign Language Using Tensorflow [7]	2.659	Public Dataset	12	Identical dataset referenced by the author could not be located.
Integrating the CNN Model with the Web for Indonesian Sign Language (BISINDO) Recognition	312	Public Dataset	26	None

Table 11 reveals significant variations among the characteristics of datasets employed in related research on BISINDO recognition. This study utilizes a public dataset containing 312 images across 26 sign classes. This dataset size is relatively small compared to most listed prior studies, several of which utilized thousands to tens of thousands of images, often sourced from private collections [7], [11], [22]. Employing a public dataset in this study, similar to [7], potentially enhances the reproducibility of the research findings, although the specific dataset used by [7] could not be definitively identified. The number of classes (26) addressed in this study falls within the range commonly investigated by previous studies (10-37 classes).

2) Model Performance comparison: As part of the evaluation process, the performance of the model developed in this study was compared with results from previous related studies. Table 12 provides a comprehensive summary of this comparison across several aspects, including the methods employed, recognition type (e.g., static/dynamic), input dimensions, training and validation accuracies (where reported), and the presence or absence of user interface (UI) integration among this study and the referenced works.

TABEL XII
COMPARISON OF MODEL PERFORMANCE WITH PREVIOUS RESEARCH

Reference Study	Method	Recognition Type	Model Input Dimension	Train Acc (%)	Val Acc (%)	UI Integration
BISINDO (Bahasa Isyarat Indonesia) Sign Language Recognition Using CNN and LSTM [11]	CNN and LSTM	Dynamic	100 × 89	96	Not mentioned	No
Indonesian Sign Language Recognition using YOLO Method [22]	CNN	Dynamic	3024×3024; 640×640	100	Not mentioned	No
Indonesia Sign Language Recognition using Convolutional Neural Network [14]	CNN	Static	60 × 60	99,48	98,39	No
Convolutional Neural Network (CNN) for Image Classification of Indonesia Sign Language Using Tensorflow [7]	CNN	Static	150 × 150	96,80	100	No

Integrating the CNN Model with the Web for Indonesian Sign Language (BISINDO) Recognition	CNN	Static	256 × 256	94,02	97,44	Yes

Table 12 contextualizes this research within the landscape of related studies. Employing a CNN method for static sign recognition, analogous to approaches in [14] and [7], this study achieved a training accuracy of 94.02% and a validation accuracy of 97.44%. While the training accuracy is marginally lower than that reported in some studies (>96%), the validation accuracy (97.44%) demonstrates competitive generalization performance compared to [14] (98.39%) and [7] (100%).

It is important to note, however, that direct accuracy comparisons across studies are inherently limited due to significant differences in the datasets used (as detailed in the Table 11), variations in the number of classes and input dimensions, and differing focuses on static versus dynamic signs [11], [22]. A unique contribution of this study, setting it apart from the compared works, is the successful integration of the model into a web-based user interface, which demonstrates the practical application potential of the developed model.

IV. CONCLUSION

Based on the research conducted, it is concluded that the developed Deep Learning (DL) model effectively recognizes static images containing hand gestures for the BISINDO alphabet. This is evidenced by the model's high accuracy, achieving 94.02% on the training data and 97.44% on the validation data. Furthermore, evaluation metrics including the confusion matrix, precision, recall, and F1-score also yielded strong results, with respective average macro-scores of approximately 98%, 97%, and 97.12%. Additionally, functional and latency testing results confirm that the web-integrated DL model functions correctly, is readily accessible, and provides rapid recognition results. Although this research successfully developed an accurate DL model and integrated it into an accessible web recognition application, the study was limited to the 26 alphabet classes of BISINDO and utilized a dataset smaller than those in some prior studies. Furthermore, direct performance testing involving users with disabilities was not performed. Consequently, subsequent development needs to overcome limitations related to data scope, dataset variability, and end-user validation for real-world applicability.

REFERENCES

[1] *World Report on Hearing*, 1st ed. Geneva: World Health Organization, 2021.

[2] L. Arisandi and B. Satya, "Sistem Klarifikasi Bahasa Isyarat Indonesia (Bisindo) Dengan Menggunakan Algoritma Convolutional Neural Network," *Jurnal Sistem Cerdas*, vol. 5, no. 3, pp. 135–146, Dec. 2022, doi: 10.37396/jsc.v5i3.262.

[3] Kemendikbud, "Tunarungu." 2023. [Online]. Available: https://kbbi.kemdikbud.go.id/entri/tunarungu

[4] BPS, *Hasil Long Form Sensus Penduduk 2020*. Badan Pusat Statistik Indonesia, 2023. [Online]. Available: https://www.bps.go.id/id/publication/2023/01/27/ffb5939b4393e5b1146a9b91/hasil-long-form-sensus-penduduk-2020.html

[5] Kemendikbud, "Kamus SIBI," *Kementerian Pendidikan dan Kebudayaan*. Dec. 2020. Accessed: Jan. 22, 2024. [Online]. Available: https://pmpk.kemdikbud.go.id/sibi/

[6] Pusbisindo, "Tentang Pusat Bahasa Isyarat Indonesia." [Online]. Available: https://pusbisindo.org/tentang-kami

[7] O. Kembuan, G. Caren Rorimpandey, and S. Milian Tompunu Tengker, "Convolutional Neural Network (CNN) for Image Classification of Indonesia Sign Language Using Tensorflow," in *2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS)*, Manado, Indonesia: IEEE, Oct. 2020, pp. 1–5. doi: 10.1109/ICORIS50180.2020.9320810.

[8] I. A. Adeyanju, O. O. Bello, and M. A. Adegboye, "Machine learning methods for sign language recognition: A critical review and analysis," *Intelligent Systems with Applications*, vol. 12, p. 200056, Nov. 2021, doi: 10.1016/j.iswa.2021.200056.

[9] M. Alaghband, H. R. Maghroor, and I. Garibay, "A survey on sign language literature," *Machine Learning with Applications*, vol. 14, p. 100504, Dec. 2023, doi: 10.1016/j.mlwa.2023.100504.

[10] M. Madhiarasan and P. P. Roy, "A Comprehensive Review of Sign Language Recognition: Different Types, Modalities, and Datasets." arXiv, Apr. 2022. doi: 10.48550/arXiv.2204.03328.

[11] A. Aljabar and S. Suharjito, "BISINDO (Bahasa Isyarat Indonesia) Sign Language Recognition Using CNN and LSTM," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 5, pp. 282–287, 2020, doi: 10.25046/aj050535.

[12] F. Alrowais, S. S. Alotaibi, S. Dhahbi, R. Marzouk, A. Mohamed, and A. Mustafa Hilal, "Sign Language Recognition and Classification Model to Enhance Quality of Disabled People," *Computers, Materials & Continua*, vol. 73, no. 2, pp. 3419–3432, 2022, doi: 10.32604/cmc.2022.029438.

[13] N. Adaloglou *et al.*, "A Comprehensive Study on Deep Learning-Based Methods for Sign Language Recognition," *IEEE Transactions on Multimedia*, vol. 24, pp. 1750–1762, 2022, doi: 10.1109/TMM.2021.3070438.

[14] S. Dwijayanti, H. -, S. I. Taqiyyah, H. Hikmarika, and B. Y. Suprapto, "Indonesia Sign Language Recognition using Convolutional Neural Network," International Journal of Advanced Computer Science and Applications, vol. 12, no. 10, 2021, doi: 10.14569/IJACSA.2021.0121046.

[15] A. N. Sihananto, E. M. Safitri, Y. Maulana, F. Fakhruddin, and M. E. Yudistira, "Indonesian Sign Language Image Detection Using Convolutional Neural Network (CNN) Method," Inspiration: Jurnal Teknologi Informasi dan Komunikasi, vol. 13, no. 1, pp. 13–21, May 2023, doi: 10.35585/inspir.v13i1.37.

[16] S. A. Sanjaya and H. Faustine Ilone, "BISINDO Sign Language Recognition: A Systematic Literature Review of Deep Learning Techniques for Image Processing," Indonesian Journal of Computer Science, vol. 12, no. 6, Dec. 2023, doi: 10.33022/ijcs.v12i6.3539.

[17] R. Borman, B. Priopradono, and A. Syah, "Klasifikasi Objek Kode Tangan pada Pengenalan Isyarat Alphabet Bahasa Isyarat Indonesia (BISINDO)," SNIA (Seminar Nasional Informatika dan Aplikasinya), vol. 3, 2019, [Online]. Available: https://snia.unjani.ac.id/web/index.php/snia/article/view/87

[18] A. Noer, "Bahasa Isyarat Indonesia (BISINDO) Alphabets." Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/achmadnoer/alfabet-bisindo

[19] V. R. Joseph, "Optimal Ratio for Data Splitting," Statistical Analysis, vol. 15, no. 4, pp. 531–538, Aug. 2022, doi: 10.1002/sam.11583.

[20] C.-H. Lin, C. Kaushik, E. L. Dyer, and V. Muthukumar, "The good, the bad and the ugly sides of data augmentation: An implicit spectral regularization perspective," 2022, arXiv. doi: 10.48550/ARXIV.2210.05021.

[21] A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Zou, "Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild," 2019, arXiv. doi: 10.48550/ARXIV.1906.02569.
[22] S. Daniels, N. Suciati, and C. Fathichah, "Indonesian Sign Language Recognition using YOLO Method," IOP Conf. Ser.: Mater. Sci. Eng., vol. 1077, no. 1, p. 012029, Feb. 2021, doi: 10.1088/1757-899X/1077/1/012029t