Journal of Engineering Science and Technology
Vol. 21, No. 4 (2026) 1016 - 1035

HYBRID RANDOM FOREST AND LONG SHORT-TERM MEMORY FOR REAL-TIME SIGN LANGUAGE CLASSIFICATION

ROMY BUDHI WIDODO[1,*], OLFAT HARITS ALATAS[1], MOCHAMAD SUBIANTO[2], JOSEPH DEDY IRAWAN[3]

[1]Ma Chung Human-Machine Interaction Research Center, Informatics Engineering, Universitas Ma Chung, Jalan Villa Puncak Tidar N-1, Malang, 65151, Indonesia
[2]Informatics Engineering, Universitas Ma Chung, Jalan Villa Puncak Tidar N-1, Malang, 65151, Indonesia
[3]Informatics Engineering, Faculty of Industrial Technology, National Institute of Technology Malang, Jl. Raya Karanglo km.2, Malang, Indonesia
*Corresponding Author: romy.budhi@machung.ac.id

Abstract

This study proposes a new hybrid approach for Indonesian Sign Language (BISINDO) recognition using camera input on a Raspberry Pi. The system integrates Random Forest (RF) for static gesture classification and Long Short-Term Memory (LSTM) for dynamic gesture recognition. Feature extraction was conducted using MediaPipe to obtain 21 hand landmarks in three-dimensional coordinates. This study utilizes numbers, letters, and word choices that frequently appear in international corpus lists, which are classified by machine learning models into distinct categories. Experimental results show that the RF model achieved an accuracy of 97.9%, while the LSTM model reached 96.5%. Real-time testing on the Raspberry Pi demonstrated low latency and stable recognition, confirming the system's feasibility in practical applications. The novelty lies in combining a classical machine learning method with a deep learning model in a hybrid design, deployed on a portable embedded device. This research contributes to the development of inclusive communication technology for the Deaf community.

Keywords: BISINDO, Hybrid model, LSTM, Random forest, Sign language recognition.

1. Introduction

1.1. Background

Indonesian Sign Language (BISINDO) is the main form of communication used by people who are deaf or hard of hearing in Indonesia. As a sign language, BISINDO is conveyed through a combination of hand movements, facial expressions, and body position, so it is often difficult for the general public who are not accustomed to using it to understand. This condition creates communication barriers between people with disabilities and the surrounding social environment. In Indonesia, there are two sign language systems, namely the Indonesian Sign Language System (SIBI) and the Indonesian Sign Language (BISINDO).

Several studies have demonstrated the effectiveness of using machine learning algorithms in classifying hand movements. First, the Random Forest (RF) and Long Short-Term Memory (LSTM) algorithms are used to recognise static hand signals based on geometric features or the results of image extraction from the camera. A study by Fadlilah et al. [1] successfully developed a BISINDO basic signal recognition system using a camera and Raspberry Pi, which can translate letter and number gestures into text displayed in real-time on the monitor screen. Nevertheless, this study has not involved the translation of words that are classified as dynamic.

Second, the LSTM model, a component of the Recurrent Neural Network (RNN) architecture, has proven effective in handling sequential data. A study by Aljabar and Suharjito [2] combined CNN and LSTM to recognise BISINDO cues in real-time using desktop devices, achieving an accuracy of up to 96% in the recognition of certain words. Third, research conducted by Alexander et al. [3] showed that the RF algorithm has superior performance compared to other classification models, such as K-Nearest Neighbour (KNN), Support Vector Machine (SVM), and Decision Tree in recognising BISINDO sign language gestures. RF achieved accuracy, precision, F1-score, and recall values of 97.9% in the test data, as well as 84% precision in real-time testing, making it the most stable and accurate model in static gesture classification.

However, in this study, static gestures and dynamic gestures have not been mapped at the time of classification. Therefore, this study aims to answer the following research questions: (1) How to design a hybrid machine learning architecture that combines RF and LSTM to effectively recognize static and dynamic sign language?; (2) To what extent does this hybrid approach improve classification accuracy compared to single-model methods found in previous literature?; and (3) Is the system feasible to implement on embedded devices with limited resources, especially the Raspberry Pi, while still maintaining real-time performance?

The novelty of this work lies in 1) Proposing a hybrid RF-LSTM model for BISINDO, with RF handling static gestures and LSTM managing dynamic gestures; 2) Deploying the hybrid model on Raspberry Pi, demonstrating real-time low-latency performance; 3) Offering an inclusive, portable, and low-cost solution for communication between deaf and hearing communities. This consideration is based on our previous study by Alexander et al. [3], which found that Random Forest does not optimally handle the sequential or temporal data contained in dynamic gestures. Additionally, references suggest that the LSTM model can recognise the sequential movement patterns typical of dynamic gestures.

1.2. Related works

Research related to BISINDO exhibits diversity, as seen in Fadlilah et al. [1], who proposed a BISINDO translator based on Raspberry Pi; however, it only handled a limited vocabulary. Aljabar and Suharjito [2] applied CNN-LSTM for BISINDO recognition on a PC, achieving good accuracy but without portable deployment. Alexander et al. [3] examined machine learning approaches for BISINDO using cameras, focusing on classification in desktop settings and utilizing RF to classify dynamic words. Other works investigated alphabet recognition in Borman et al. [4] and Pujiati [5], sign language learning applications by Assa et al. [6], and Nugraheni et al. [7] Compare and optimise the use of BISINDO and SIBI. Lugaresi et al. [8] introduced MediaPipe for landmark extraction, which has been widely adopted in sign language research.

Breiman [9] and Biau and Scornet [10] laid the foundation for RF, while Graves [11] and Van Van Houdt et al. [12] established LSTM. Researchers have utilized LSTM in classifying sign language, as demonstrated by Gu et al. [13], who employed LSTM and Transformer for American Sign Language (ASL) translation, utilizing IMU and EMG sensors. However, the use of sensors is considered less practical in everyday, real-time applications. Then Huang and Chouvatut in [14] translated the Argentine sign language by utilizing the ResNet architecture to extract features, followed by an LSTM to obtain long sequence features. However, in this case, the research has not been implemented on hardware.

Similarly, Kumari and Anand [15] state that the hybrid model, which combines CNN with MobileNetV2 and LSTM architectures, yields good accuracy. Moreover, a hybrid model was highlighted by Baihan et al. [16], which combined CNN, Self-Attention, and LSTM to detect dynamic features. These show that hybrid models with LSTMs provide increased accuracy. The use of bidirectional LSTM was applied to record gestures and replay sounds based on gesture movements through a smartphone by Rafiq et al. [17]. This demonstrates that LSTM is highly effective in processing long sequence features. This is supported by Tan et al. [18] which provides a comprehensive overview of deep learning approaches.

Tan et al. notes that although model architectures such as LSTM and Transformer have significantly improved accuracy and struck a balance between computational complexity and speed of recognition; remains a critical challenge in the development of assistive technologies for the deaf community. In addition, Papatsimouli et al. [19] conducted a systematic survey on real-time sign language translators, which specifically emphasized integration with IoT and edge computing technologies, as well as emphasizing the need to develop lightweight models that can operate on portable devices with limited resources.

This proposed study builds upon a continuing study by Alexander et al. [3], in which a prototype was developed on the desktop that successfully compared four types of classifier models from Machine Learning, namely RF, KNN, SVM, and Decision Tree. The models used were successful in identifying BISINDO sign language, although they have shortcomings and are still longer compared to the typing method. The best model performance is achieved by RF, with accuracy, precision, F1, and recall values of 97.7% obtained from model testing, and 84% precision in real-time. It was able to achieve the standard of the shortest duration in classifying gestures, which is 34.6 words per minute. Figure 1 shows research

by Alexander et al. [3], which is a desktop-based application. A Subject sitting in a chair in front of a camera and a computer to classify the hand gesture.

Fig. 1. Our previous study in Alexander et al. [3], when subjects experiment with the application.

2. Materials and Methods

This study uses machine learning theory, where the initial process is 1) data collection; 2) data processing (pre-processing, including data augmentation); 3) training to obtain a classification model; 4) Deployment to the device. In step 1, which involves data collection, a Logitech C270 USB web camera is required. This is an HD 720p webcam with automatic light correction. Meanwhile, the frameworks used are MediaPipe, the Python programming language, and the OpenCV library. In steps 2 and 3 for data processing and training, the apparatus was PC hardware with Windows 11, Intel i7 12th gen, 32GB RAM, NVIDIA GeForce RTX 3060 GPU, and 1TB SSD were used; TensorFlow framework; some popular open-source libraries such as Imgaug, Joblib, Pickle, Matplotlib, Pandas, Scikit-learn, Numpy; and the Collections module. Step 3 involves deploying the system using a Raspberry Pi 4 Model B with 8GB of RAM.

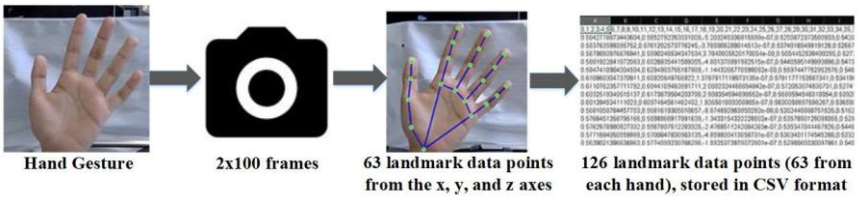
2.1. Data collection

The researchers focus on the ethical considerations of assistive technology deployment through participatory design. From the design stage the researchers engage Deaf individuals who help in providing sign language advice and training. The data collection process was conducted in two separate shooting sessions. At each session, the subject is asked to perform a specific gesture, which is then recorded as a frame image of 100 images per gesture.

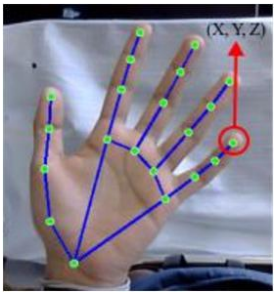
The images are then used as training data to train the Machine Learning model. The flow of this data collection process is illustrated in Fig. 2(a). Once the shooting process is complete, the obtained images will be further processed to extract hand coordinate points using MediaPipe Holistic. Although MediaPipe Holistic provides pose detection, face, and hand detection simultaneously, this study utilized only the hand detection part. The system automatically detects the position of the hand and represents it in the form of three-dimensional coordinates of landmark points.

The coordinates consist of x, y, and z values, where x indicates the horizontal position (from left to right), y indicates the vertical position (from top to bottom), and z represents the depth or relative distance of a point from the camera. This z-value does not come from the depth camera but is the result of a built-in estimate by the MediaPipe model based on 2D imagery. Figure 2(b) shows the results of the

visualisation Hand Landmark Detection, where each node represents a single point on the structure and stores the value of the three-dimensional coordinates.



(a)



(b)

Fig. 2. (a) The flow of the data collection process, (b) Three-dimensional coordinates of each node of the landmark.

The study utilized forty words that frequently appear in the Corpus of Contemporary American English (COCA), as listed on www.wordfrequency.info [20]. Besides that, there are eleven numbers, namely: 0 to 10, and the alphabet, namely A to Z. Therefore, a total of 77 sign languages need to be classified. The researchers classify movements into two categories: static and dynamic gestures. The difference between the two lies in the presence or absence of hand movements when using gestures. Static gesture refers to a motionless/static hand when performing the gesture, whereas a dynamic gesture involves a waving motion of the hand when the gesture is made. Number 0 to 10 are classified as static gestures, while alphabet A to Z, only R is classified as a dynamic gesture. Table 1 shows the forty-word choices in this study based on COCA.

Table 1. Forty daily corpus list.

Word	Word	Word
Category: static gestures		
water	this	bread
you	that	I/me
up	so	like
or	if	who
down	work	year
learn	outside	but
can	eat	for
from	drink	she
only	up/go up	go

Table 1 (Continued). Forty daily corpus list.

Word	Word	Word
Category: dynamic gestures		
will	take	what
say	they	we
and	in	own/have
people	all	know/ understand
Down/go down		

2.2. Data processing

In the pilot testing stage, with a total of 200 images per gesture in subchapter 2.1, the LSTM model training process to recognise dynamic gestures shows symptoms of overfitting. This is due to the limited amount of data, which is not enough to capture the variation of dynamic hand movements in a representative manner. To overcome these problems, data augmentation is employed, which involves artificially expanding datasets by modifying the original images through visual transformations. Data augmentation has become a widely adopted strategy in computer vision to improve model generalisation and reduce overfitting [21].

This process aims to enrich the training data and improve the generalisation capabilities of the LSTM model to dynamic gestures. Augmentation is carried out using the image library. The applied augmentations include Gaussian blur, noise addition, brightness and contrast changes, and geometric transformations such as rotation, translation, scale (zoom in/out), shearing, and flipping, as done by Papatsimouli et al. [19].

The purpose of this variation is to simulate the varying conditions of real shooting, thereby making the trained model more robust against changes in hand position, angle of view, or image quality. Each image in the original dataset was augmented five times, resulting in an additional 30,000 images per gesture. This process was performed for each category of dynamic gestures that were previously separated from the RF model, which is unable to accurately detect those dynamic gestures.

Figure 3(a) illustrates the state of the dataset before augmentation, which remains limited, and Fig. 3(b) shows a significant increase in data volume after the augmentation process. With the rise in the amount of augmented data, it is expected that the LSTM model can be better trained and able to generalise dynamic gestures more effectively. This dataset of augmented results is then reused in the feature extraction process using MediaPipe.

(a)

(b)

Fig. 3. (a) The dataset before augmentation, (b) The dataset after augmentation.

2.3. Training to obtain a classification model

In this subchapter, there are two steps in the data training process to obtain the Random Forest and LSTM models. The processed data will be separated into 80% training data and 20% test data.

2.3.1. Random forest model training process

Before training the LSTM model to recognise dynamic gestures, the system first uses the RF model to classify static gestures. This model is designed to recognise gestures that do not involve movement in time (such as numbers, letters, and static words). The RF model is developed using default parameters, with the following architecture: n_estimators (number of decision trees used in the ensemble) = 10-200 to evaluate performance; Criterion (split selection criteria on each node based on Gini index) = "gini"; max_depth (tree depth is not limited) = None; min_samples_split (at least two samples are required to split nodes) = 2; min_samples_leaf (minimum number of the samples needed to be present at a leaf node) = 136; max_features (the number of features considered in each split is taken from the squareroot of total features) = "sqrt". Figure 4 illustrates the RF model used in our study.

Once the model has been successfully trained, the results are saved in a file named RandForest.pkl. This file contains the model object that has been trained and is ready to use back in the sign language classification system in real-time. This Random Forest model is explicitly used to recognise static gestures and then combined with the LSTM model to handle dynamic gestures, so that the system can classify different types of BISINDO gestures more comprehensively.

2.3.2. LSTM model training process

After the hand landmark feature is successfully extracted, the data is used to train a classification model based on LSTM. This model is designed to learn the time sequence of hand coordinates that represent dynamic gestures, making them

particularly relevant to recognising the sequence of hand movements in BISINDO. The dataset used has been augmented. It is converted into the form of dimensional sequences (samples, 30, 126), i.e., 30 frames per gesture with 126 features (x, y, z coordinates of 21 points in each hand). The data was normalized using StandardScaler, and its labels were encoded using LabelEncoder, then converted to a one-hot encoding format for multiclass classification purposes. The LSTM model is built using a two-tiered architecture, with dropout layers to prevent overfitting, and two dense layers at the end. The architecture is described in Fig. 5.

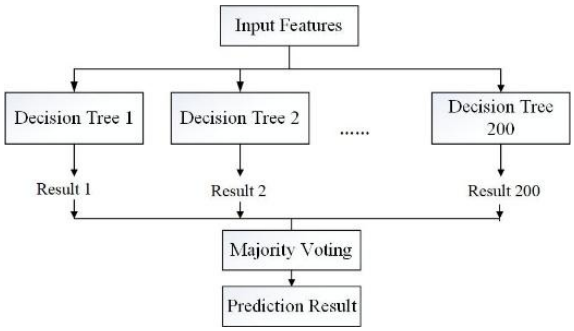


Fig. 4. The architecture of the Random Forest model.

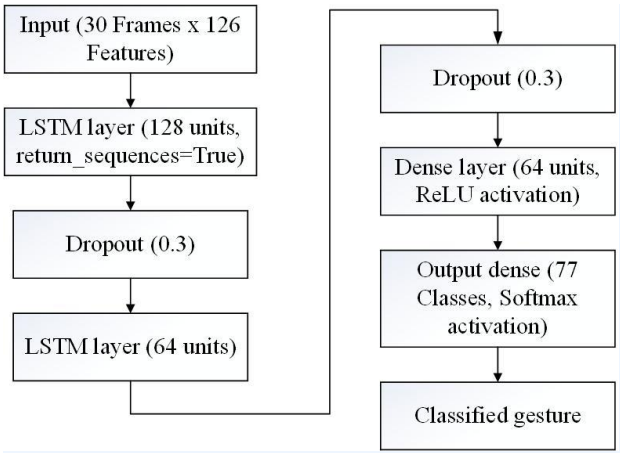


Fig. 5. The architecture of the **LSTM model.**

The model is trained using the softmax activation function and the categorical cross-entropy loss function, with the Adam optimization algorithm. To maximize training results, EarlyStopping is used to stop training early if validation performance does not improve, and ModelCheckpoint is used to store the best model weights based on val_loss values. The model was built in stages, starting with two layers of LSTMs, each with 128 and 64 units, respectively, interspersed with a Dropout of 30% to prevent overfitting. After that, the output is routed to the Dense layer containing 64 neurons with ReLU activation, and terminates with the Dense output layer, which uses softmax activation for multiclass classification accordingly to the number of categories in y_categorical.

The training process is run for up to 100 epochs with a batch size of 32 and validation on X_val data. To improve training stability, the researchers applied EarlyStopping, which automatically stops training if there is no improvement in val_loss for 10 epochs, as well as ModelCheckpoint, which saves the best weights in the best_model_lstm.h5 file. Training results show excellent performance, with the accuracy testing reaching 98%. Based on the classification report, most of the gestures have a precision, recall, and F1-score value above 0.90. This demonstrates that the model can accurately classify dynamic gestures. Training of this model generates some essential files:

1. best_model_lstm.h5 is the best LSTM model kept for training and used in hand tracking systems for real-time classification of dynamic gestures;
2. label_encoder.pkl is used to convert the numerical output of the model into the shape of the text label (e.g., "CAN", "WHAT", "DOWN/GO DOWN"); and
3. scaler.pkl is used to normalise the coordinate input when the system is running, so that the value scale corresponds to the conditions when the model is trained.

The best_model_lstm.h5 model will be loaded by the hand tracking system during the first run. When the user demonstrates gestures, the system captures 30 frames, extracts the coordinate points of the hand using MediaPipe, and performs normalization using a scaler.pkl file, and then send the data to the LSTM model for prediction. The model's output is a probability class, which is converted back to the corresponding gesture name using label_encoder.pkl and displayed as the result of dynamic gesture recognition. In LSTM model training, several hyperparameters are used to control the learning process and impact the model's final performance. These hyperparameters are determined before training begins and remain unchanged throughout the training process. The following is an explanation of each hyperparameter used in this study:

- input_shape = (30, 126)

  The input model consists of a sequence of 30 frames, each with 126 features. It reflects 30 steps of time (frame), each containing 21 landmark points on two hands (left and right), with three coordinates (x, y, z) for each point.

- LSTM (128, return_sequences=True)

  The function of this step is to capture temporal patterns. The first LSTM layer has 128 units of memory and returns a sequential output for each timestep, allowing it to be passed on to the next LSTM. Size 128 was chosen to balance between learning capacity and computational efficiency.

- LSTM(64)

  The purpose of this step is to gain a deeper understanding of sequential relationships. The second LSTM layer has 64 units. It does not return a sequence because the results are directly passed to the dense layer for classification.

- Dropout(0.3)

  A 30% dropout ratio is applied after each LSTM layer to prevent overfitting by ignoring a portion of the unit during training.

- Dense(64, activation='replayed')

A fully connected layer with 64 neurons and a ReLU activation function is used to strengthen non-linearity before the final classification stage or transforms features into discriminative representations.

- Dense(output, activation='softmax')

  The function of the output dense layer is to produce a probability distribution for gesture classification. The output layer consists of neurons equal to the number of classes, utilizing the softmax activation function to convert the scores into the probabilities of each class.

- optimizer='adam'

  Adam's optimisation algorithm was chosen because it is efficient and adaptive to weight updates.

- loss='categorical_crossentropy'

  This loss function is used for multiclass classification problems with one-hot encoding labels.

- metrics=['accuracy']

  Accuracy is used as a key metric to evaluate model performance during training and validation.

- epochs=100

  The maximum training runs for 100 epochs. However, training can be stopped early by EarlyStopping.

- batch_size=32

  Each iteration of the training was carried out in batches of 32 samples.

- EarlyStopping(patience=10)

  Training will stop automatically if there is no decrease in validation loss for 10 consecutive epochs.

- ModelCheckpoint(monitor='val_loss')

  The best model is saved based on the lowest validation loss value during training and is stored in a best_model_lstm.h5 file.

These hyperparameters are designed to strike a balance between model capacity, generalization, and training efficiency, especially considering that the system will be implemented on devices such as the Raspberry Pi, which have limited resources.

2.4. Deployment of the device

This gesture detection system is executed in real-time using a Logitech C270 USB webcam connected to a Raspberry Pi 4 Model B. When the camera is active and the user's hand is successfully detected, the system extracts feature from the hand landmarks obtained using MediaPipe Holistic. Each hand earns 21 points, and coordinates are defined in three axes (x, y, z), so each frame includes a total of 126 features for both hands. The data from each of these frames is then stored in two separate buffers: the static buffer and the sequence buffer. A static buffer works to store a single frame. It is directly used as input for the RF model, which is designed to recognize gesture patterns that are static or immutable between frames.

On the other hand, the sequence buffer stores a network of 30 frames in sequence, which is then used as input for the LSTM model to detect movement patterns in dynamic gestures. Decision-making about which model is used to process feedback is done adaptively by the system. Suppose the sequence buffer has successfully collected 30 frames and has passed a specific delay time since the last LSTM prediction. In this case, the system enables the LSTM model to perform dynamic gesture classification. This delay is designed to prevent repetitive processing of the same gesture within a given period and to allow the system sufficient time to gather movement information.

However, suppose these conditions have not been met, for example. In that case, if the number of frames is insufficient or the delay has not completely elapsed, the system will use the RF model directly to predict static gestures using a single current frame. For the visualization of how the software flowchart is described, as shown in Fig. 6.

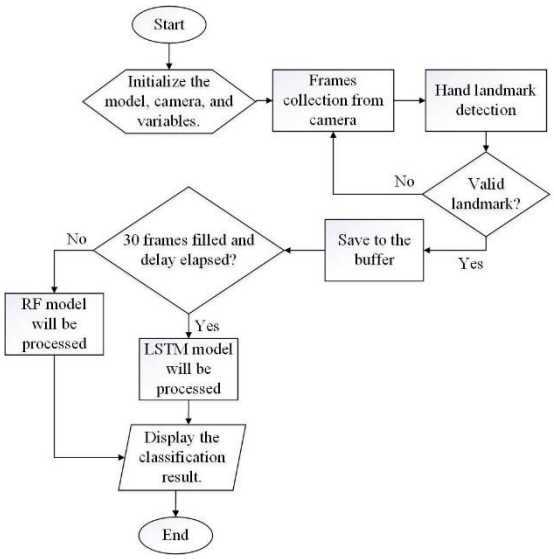


Fig. 6. Flowchart of real-time sign language classification.

The results of the gesture classification are displayed in real-time text on the monitor display. As shown in Fig. 7, the word "THAT" appears on the upper left side of the display as a result of the classification process of one of BISINDO's gestures.

In addition to the classification results in text, the system also displays scores from the two classification models used. Numbers such as "Random Forest: 188" and "LSTM: 37" reflect the number of predictions that each model has made since the system was launched. This information serves to monitor the model's contribution to the classification process: RF is used to recognise static gestures, while LSTM is used for dynamic gestures. With this model combination approach, the system can handle different types of gestures flexibly and automatically based on the type of input received from the camera.

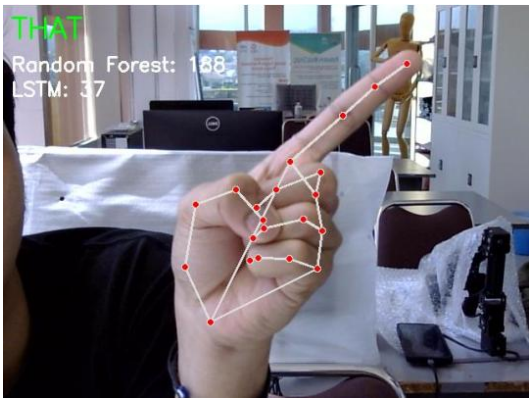


Fig. 7. Real-time implementation output example.

3. Experiment Results

This section will be divided into two parts: the results of the model evaluation and the results of the real-time assessment.

3.1. Results of the model evaluation

The model evaluation was conducted to assess the system's performance in classifying BISINDO gestures, specifically in the categories of static and dynamic gestures, using the RF model and the LSTM model, respectively. Based on the test results, the RF model used in Alexander et al.'s [3] study showed remarkable accuracy with a value of 0.98 for all primary evaluation metrics, namely precision, recall, and F1-score. This shows that the RF model is highly reliable in recognising static gestures whose shape tends to be consistent, such as those in Table 2, including the numbers 0 to 10 and the alphabet A to Z, except for the letter R.

Table 2. Static gestures for random forest model testing.

Word	Word	Word
water	this	bread
you	that	I/me
up	so	like
or	if	who
down	work	year
learn	outside	but
can	eat	for
from	drink	she
only	up/go up	go
numbers 0 to 10		
alphabets A to Z (except R)		

The evaluation of the RF model was conducted by referring to the confusion matrix generated after the model was applied to the BISINDO gesture dataset; the confusion matrix can be viewed at https://bit.ly/4syqoBM. In general, the evaluation results show that the RF model exhibits good classification performance against most static gestures. This is evident from the dominance of the correct

prediction value in the diagonal element of the confusion matrix, indicating a high level of accuracy. Alphabetic gestures such as “A”, “B”, “C”, to “Z” are consistently recognisable, and gestures with distinctive hand shapes such as “EAT”, “DRINK”, or “I/ME” are also accurately classified by the model. This shows that RF is quite reliable at recognising strong and consistent static visual patterns between users.

However, some gestures often experience errors in classification, especially those with highly similar hand shapes. For example, alphabetic gestures such as “I” and “J” or “V” and “U” tend to be difficult for models to distinguish because they have only minor differences in the representation of finger positions. In addition, numerical gestures, such as “1”, “7”, and “9,” also indicate classification confusion due to high visual similarity, particularly when performed by different subjects or in less-than-ideal lighting conditions. The misclassification that occurs in the confusion matrix is evenly distributed across the various gestures and is not concentrated in just one or two classes. This indicates that the RF model exhibits fairly even resistance to static gestures although there is still room for improvement in certain gestures that exhibit higher error consistency.

It is also important to note that with the transfer of dynamic gestures to the LSTM model, the burden of classification of gestures becomes more evenly distributed. Random Forest can focus more on static gestures that suit the character, while dynamic gestures that require the context of time and movement sequence are more precisely handled by sequential models such as LSTM. This approach not only enhances classification accuracy but also improves the overall effectiveness of the system.

Overall, the evaluation indicates that the RF model has significant potential for use in static gesture classification in BISINDO sign language, particularly when supported by balanced data distribution, adequate lighting, and a consistent labelling process. This model provides a solid foundation for integrating responsive and accurate camera-based gesture recognition systems, particularly in environments that require real-time user interaction. Meanwhile, Table 3 shows that the LSTM model developed in this study was used to detect dynamic gestures.

Table 3. Dynamic gestures for LSTM model testing.

Word	**Word**	**Word**	**Word**
will	take	what	say
they	we	and	in
own/have	people	R	all
know/ understand	down/go down		

The confusion matrix of the LSTM model’s prediction results, shown in https://bit.ly/4uiMOZu, indicated that the model accurately classified the gestures according to their original labels. Some gestures, such as TAKE, WHAT, SAY, IN, AND, WE, THEY, OWN/HAVE, R, ALL, KNOW, and DOWN, were perfectly recognised with 100% accuracy. Overall, the model’s performance is good because the error distribution is minimal and does not affect the majority of correct predictions.

The hybrid model achieved an overall accuracy of 0.98, with balanced precision and recall values, as shown in Table 4.

Table 4. Classification report results from the RF and LSTM models.

Model	f1	Recall	Precision	Accuracy
Random Forest	0.98	0.98	0.98	0.98
LSTM	0.98	0.98	0.98	0.98

Figure 8 illustrates the accuracy and loss graphs during the LSTM model training process over 42 epochs.

The training accuracy curve (Train Accuracy) shows a consistent and steady increase, approaching 100%. In contrast, the validation accuracy curve (Val Accuracy) tends to follow a similar pattern, albeit with slight fluctuations. This indicates that the model can effectively learn patterns from the training data and exhibits stable performance on the validation data.

Meanwhile, the loss chart suggests that the Train Loss decreased sharply at the beginning of training and continues to approach zero, indicating that the prediction errors in the training data have been significantly suppressed. The validation loss also decreases initially but begins to fluctuate after a certain point, exhibiting mild overfitting symptoms. This occurs when the model starts to over adjust to the training data, thereby decreasing its ability to generalize.

To overcome this issue, an early stopping technique with the parameter patience=10 is used, which automatically stops training if there is no improvement in val_loss for 10 consecutive epochs. Thus, although training is scheduled for up to 100 epochs, the best models are retained at the time val_loss reaches its minimum, which typically occurs around the 33rd to 35th epoch, before significant fluctuations begin to appear.

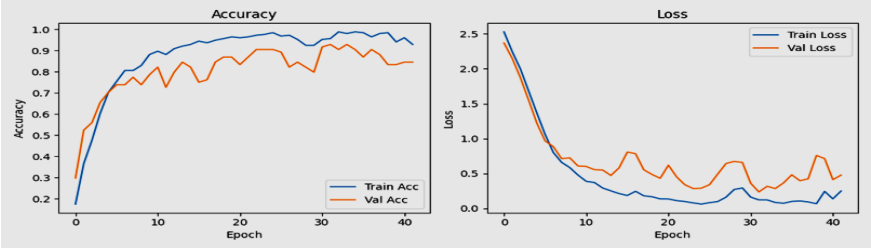


Fig. 8. Accuracy and loss graph during model evaluation.

3.2. Results of the real-time evaluation

Once the best LSTM model is obtained through the training process, the next stage is to evaluate the system's performance in real-time on the Raspberry Pi. The test was carried out by inviting six subjects, all of whom are students. Each subject was asked to demonstrate the entire category of gestures consisting of gestures of numbers 0-10, alphabet gesture A-Z, 14 dynamic word gestures, and 27 static word gestures.

The testing process was conducted using the Logitech C270 USB webcam mounted on the Raspberry Pi 4 Model B, as shown in Fig. 9. During the real-time testing, the subjects were positioned at a distance of approximately 1.5 meters from the webcam.

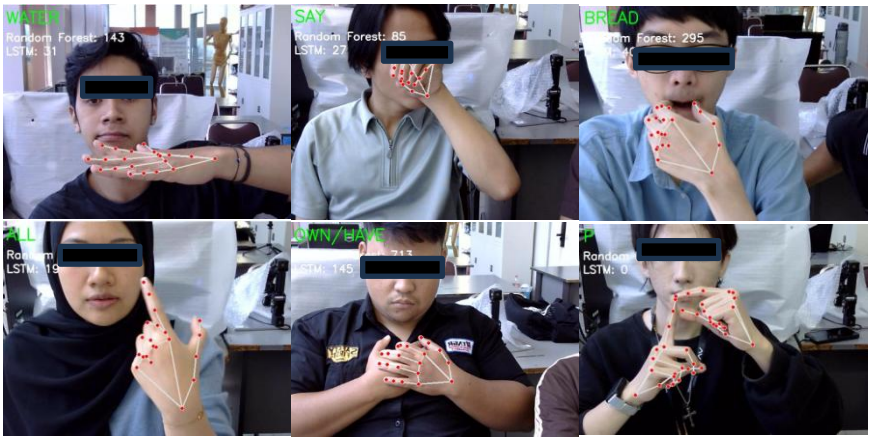


Fig. 9. Documentation of the model testing process on the subject.

For each gesture, each subject performed five trials. The data from these test results are compiled in Table 5.

Table 5. Results of real-time evaluation.

Model	Accuracy
Random Forest for static gestures	0.77
LSTM for dynamic gestures	0.34
Hybrid model for all gestures	0.69

To evaluate the resilience and applicability of the model, three statistical tests were carried out, namely: 1) The difference test in static gesture using RF method between training and real-time results, the results are in Fig. 10; 2) The difference test in dynamic gesture using LSTM method between training and real-time results, the results are described in Fig. 11; and 3) The difference test from the whole (hybrid) static and dynamic gesture during training with real-time results, the results are in Fig. 12.

In the first difference test, statistical analysis using the Wilcoxon Signed Ranks Test was carried out to measure the significance of the difference in performance from 63 classes/sample of experiments.

	Accuracy_Realtime_RF - Accuracy_Train_RF
Z	-4.434[b]
Asymp. Sig. (2-tailed)	.000

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

Fig. 10. Test Statistics of RF Train Results and Realtime RF Results.

The results of the statistical test in Table 6 show a statistically significant difference in performance between the two test scenarios ($Z = -4.434$, $p < 0.001$). Further analysis of the Ranks distribution indicates that these significant

differences lead to a decrease in performance when the system operates live. Specifically, out of a total of 63 observations, the majority of cases (N=36) showed Negative Ranks, which means that the real-time accuracy level is lower than the accuracy in the testing phase (Training/Testing), with a very dominant weight of decline (Sum of Ranks = 1427.50). In contrast, there were only 22 cases (Positive Ranks) that experienced an increase in accuracy (Sum of Ranks = 283.50), while the other 5 cases did not change (Ties).

In the second test, a statistically different test was conducted to evaluate the performance of the LSTM model specifically in 14 dynamic gesture classes. The results of the Wilcoxon Signed Ranks Test revealed a statistically significant performance degradation during real-time system implementation (Z = -3.296, p = 0.001), as shown in Table 7.

	Accuracy_Realtime_LSTM - Accuracy_Train_LSTM
Z	-3.296[b]
Asymp. Sig. (2-tailed)	.001

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

Fig. 11. LSTM Train results test statistics and LSTM real-time results.

In the second test, the analysis of the rank distribution (Ranks) shows that all tests on the 14 dynamic gestures produced Negative Ranks (N=14, Sum of Ranks weights = 105.00). This indicates that the real-time accuracy rate is absolutely lower than the accuracy in the training phase for each test case.

In the third difference test, statistical testing was carried out on the entire combined classification system which included 77 classes of gestures (static and dynamic). The Wilcoxon Signed Ranks Test showed that the application of the model in real-time resulted in a statistically significant decrease in accuracy (Z = -5.695, p < 0.001), as shown in Table 8.

	Accuracy_Realtime - Accuracy_Train
Z	-5.695[b]
Asymp. Sig. (2-tailed)	.000

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

Fig. 12. Test statistics of RF+LSTM Train results and RF+LSTM real-time results.

Analysis of the rank distribution of the third test, showed that the system experienced more performance degradation, with 50 gesture classes being on Negative Ranks (Sum of Ranks = 2444.00). Although 24 gestures recorded Positive

Ranks (Sum of Ranks = 331.00) and 3 gestures did not change (Ties = 3), the amount of the decrease value far dominated the overall performance of the system.

Although the Raspberry Pi has advantages in terms of portability and low cost, the test results show that the detection performance is reduced compared to offline program testing. One of the primary factors contributing to this is the Raspberry Pi's limited computing power, which prevents the camera display from running at high frame rates. This causes the LSTM model, which is sensitive to the sequence of motion, to be less optimal in recognizing sequential patterns promptly.

In addition to hardware technical factors, some gestures are not accurately detected due to:

- Flexibility and hand structure of each different subject,
  the majority are in the gesture of the number "9", where the subject has difficulty bending the little finger.
- Some of the gestures do have complex movement characteristics, making them difficult to imitate consistently by new subjects.
- Specific for "OWN/HAVE" and "AND" gestures, test results indicate a low detection rate. This is due to the shape of the gestures, which involve both hands stacked on top of each other, whereas MediaPipe often only detects one hand dominantly, mainly if one hand covers the other.

Real-time test results using hybrid models show that the overall detection success rate of the system is 69%, based on five attempts for each gesture by six different subjects. These results indicate that the system is working quite well; however, its performance depends significantly on the characteristics of each gesture. Some gestures such as the numbers 0–5, 7, 8, 10, as well as the alphabet such as B, E, I, K, L, M, N, O, Q, S, U, V, W, X, Y, and Z, and vocabulary such as UP, OR, DOWN, LEARN, FROM, WORK, OUTSIDE, and LIKE, indicate the level of accuracy very high detection, even reaching 100%. This is because they tend to be static, have a distinctive and consistent hand shape, and are well recognizable by MediaPipe and RF models without relying on motion sequences.

In contrast, several other gestures show low detection rates. For example, the TAKE, IF, AND, and WE gestures were completely undetectable (0%), while OWN/HAVE, I/ME, EAT, DRINK, KNOW/UNDERSTAND, and WHAT were detected at a rate of 20%. These gestures are generally dynamic or two-handed gestures, which pose challenges in the detection process. For example, OWN/HAVE and AND are performed with both hands stacked on top of each other, so MediaPipe often only captures one dominant hand. Meanwhile, gestures such as TAKE, I/ME, or EAT, while not particularly complex, can be misclassified due to inconsistent speed or form of movement between subjects.

Additionally, the detection results for alphabets such as A, R, and J are also relatively low, with accuracies of 3%, 30%, and 57%, respectively. This is likely due to the similarity of features to other gestures that cause the model to be misclassified, such as the letter A, which is often detected as a BUT or X gesture. Factors such as the instability of the camera frame when using the Raspberry Pi, the variability in movement force between subjects, and the possibility of landmarks not being fully detected were the leading causes of performance degradation in this real-time test. This confirms that the application of the system

on devices with limited resources needs to be adjusted to the model's detection and processing capabilities to provide optimal results.

To validate the proposed method, a comparative analysis with previous studies was performed. In terms of model capability, offline evaluations showed superior performance, achieving 98% accuracy for static (Random Forest) and dynamic (LSTM) gestures. These results surpass the desktop-based CNN-LSTM approach by Aljabar et al. [2], which reached 96%, and significantly outperform the LSTM-based study by Nur et al. [22] with an accuracy of 85%. The proposed method also competes with the recent findings of Kautsar et al. [23], which reported 94.67% accuracy using the LSTM architecture.

However, real-time testing on the Raspberry Pi revealed the inherent challenges of embedded implementation. The accuracy of real-time static gestures with Raspberry Pi was 77% lower than the reported accuracy when using a desktop of 84%, as reported by Alexander et al. [3]; this difference is due to the limitations of Raspberry Pi's computing resources compared to the desktop environment used in their study. Nonetheless, the overall system accuracy with Raspberry Pi of 69% is consistent with Fadlilah et al. [1], who reported an average accuracy of 69.1%. The distinguishing feature of this study is the application of dynamic gesture recognition on edge devices. Although real-time dynamic accuracy drops to 34% due to frame rate latency, the study highlights the trade-off between model complexity and hardware constraints in portable assistive technologies.

3.3. Future work

Based on the findings and limitations of this study, several concrete recommendations are proposed for further research: 1) Model optimization for edge devices: model compression techniques with post-training quantization and pruning models are expected to reduce inference time and memory usage in Raspberry Pi; 2) Multi-modal recognition of hand and facial gestures: next dataset need to include facial expression features; 3) Development to a two-way translator through the integration of the Speech-to-Text module, text can be displayed on the Raspberry Pi screen to facilitate a complete conversation between deaf and non-disabled users; 4) Apply natural language processing so that the results of hand gesture translation are not discrete per word but form natural language sentences.

4. Conclusion

This research has successfully developed a BISINDO sign language translation system based on a camera running on a Raspberry Pi. The system employed two distinct classification approaches: RF for static gestures and LSTM for dynamic gestures. The system is designed to recognize gestures from the coordinate data of hand landmarks obtained through MediaPipe Holistic. From the results of offline training and testing, both models used show excellent performance, with model accuracy reaching 98%. Real-time evaluation results indicate a significant decrease in performance, with an average detection success rate of 69%.

This decline was caused by several factors, including limitations in the Raspberry Pi's computing performance, which makes data processing less smooth, as well as unstable camera capture quality that affects the consistency of the sequence input for the LSTM model. In addition, user factors also influence this, especially when gestures require complex or two-handed coordination. In some gestures, MediaPipe

detects only one hand dominantly, which means the system does not receive complete information to accurately recognize the gestures. Some complex or less common gestures also proved challenging to imitate precisely by subjects who were not active users of sign language, thus lowering the accuracy of the results.

Acknowledgement

The authors gratefully acknowledge the financial support provided by the Directorate of Research and Community Service, Directorate General of Research and Development, under the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia for the Fiscal Year 2025 through the Fundamental Research Scheme.

Abbreviations

BISINDO	Indonesian Sign Language (Bahasa Isyarat Indonesia)
CNN	Convolutional Neural Network
COCA	Corpus of Contemporary American English
GPU	Graphics Processing Unit
KNN	K-Nearest Neighbor
LSTM	Long Short-Term Memory
RF	Random Forest
RNN	Recurrent Neural Network
SIBI	Indonesian Sign Language System (Sistem Bahasa Isyarat)
SVM	Support Vector Machine

References

1. Fadlilah, N.; Prasetyo, R.A.R.; Mahamad, A.K.; Handaga, B.; Saon, S.; and Sudarmilah, E. (2022). Modelling of basic Indonesian sign language translator based on raspberry Pi technology. *Scientific and Technical Journal of Information Technologies*, *Mechanics and Optics*, 22(3), 574-584.
2. Aljabar, A.; and Suharjito, S. (2020). BISINDO (Bahasa Isyarat Indonesia) sign language recognition using CNN and LSTM. *Advances in Science*, *Technology and Engineering Systems Journal*, 5(5), 282-287.
3. Alexander, N.; Widodo, R.B.; and Swastika, W. (2023). The use of machine learning in BISINDO sign language classification using cameras. *Proceedings of the Prosiding Seminar Nasional Universitas Ma Chung* (*SENAM*), Malang, Indonesia, 11-26.
4. Borman, R.I.; Priopradono, B.; and Syah, A.R. (2017). Classification of Hand-coded Objects in Indonesian sign language alphabet recohnition (BISINDO). *Proceedings of the Seminar Nasional Informatika dan Aplikasinya* (*SNIA*), Bandung, Indonesia, 1-4.
5. Pujiati, D. (2019). *Comparison of the Structure between the Indonesian Sign System (SIBI) and the Indonesian Sign Language* (*BISINDO*): *A Syntactic Study*. Bachelor's Thesis, Department of Languages and Indonesia Literature, Universitas Pendidikan Indonesia.
6. Assa, M.C.; Kaunang, S.T.G.; and Sugiarso, B.A. (2021). Interactive application for learning indonesian sign language. *Jurnal Teknik Elektro dan Komputer*, 10(2), 135-144.

7. Nugraheni, A.S.; Husain, A.P.; and Unayah, H. (2021). Optimizing the use of sign language with SIBI and BISINDO for deaf students at the PGMI Program of UIN Sunan Kalijaga. *Jurnal Holistika*, 5(1), 28-33.
8. Lugaresi, C. et al. (2019). Mediapipe: A framework for building perception pipelines. *arXiv*:1906.08172.
9. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.
10. Biau, G.; and Scornet, E. (2016). A random forest guided tour. *TEST*, 25(2), 197-227.
11. Graves, A. (2012). *Long short-term memory*. In Graves, A. (Ed.), *Supervised sequence labelling with recurrent neural networks*. Springer Berlin Heidelberg, 37-45.
12. Van Houdt, G.; Mosquera, C.; and Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8), 5929-5955.
13. Gu, Y.; Zheng, C.; Todoh, M.; and Zha, F. (2022). American sign language translation using wearable inertial and electromyography sensors for tracking hand movements and facial expressions. *Frontiers in Neuroscience*, 16, 962141.
14. Huang, J.; and Chouvatut, V. (2024). Video-based sign language recognition via ResNet and LSTM Network. *Journal of Imaging*, 10(6), 149.
15. Kumari, D.; and Anand, R.S. (2024). Isolated video-based sign language recognition using a Hybrid CNN-LSTM framework based on attention mechanism. *Electronics*, 13(7), 1229.
16. Baihan, A.; Alutaibi, A.I.; Alshehri, M.; and Sharma, S.K. (2024). Sign language recognition using modified deep learning network and hybrid optimization: A Hybrid Optimizer (HO) based optimized CNNSa-LSTM approach. *Scientific Report*, 14(1), 26111.
17. Rafiq, R.B.; Araib Karim, S.; and Albert, M.V.(2023). An LSTM-based gesture-to-speech recognition system. *Proceeding of the* 2023 *IEEE* 11th *International Conference on Healthcare Informatics* (*ICHI*), Houston, TX, USA, 430-438.
18. Tan, S.; Khan, N.; An, Z.; Ando, Y.; Kawakami, R.; and Nakadai, K. (2024). A review of deep learning-based approaches to sign language processing. *Advanced Robotics*, 38(23), 1649-1667.
19. Papatsimouli, M.; Sarigiannidis, P.; and Fragulis, G.F. (2023). A survey of advancements in real-time sign language translators: Integration with IoT technology. *Technologies*, 11(4), 83.
20. WordFrequency. (n.d.). Word frequency: Based on one billion word COCA corpus. Retrieved September 18, 2023, from https://www.wordfrequency.info/samples.asp.
21. Amarù, S.; Marelli, D.; Ciocca, G.; and Schettini, R. (2023). DALib: A curated repository of libraries for data augmentation in computer vision. *Journal of Imaging*, 9(10), 232.
22. Nur, S.; Assyifa, A.N.; and Nurjannah, H. (2023). Development of Indonesian sign language translator application (BISINDO) using long-short term memory method. *EDUSAINTEK: Jurnal Pendidikan*, *Sains dan Teknologi*, 11(1), 13-30.
23. Kautsar, M.D.; Hariono, A.M.; and Akmal, R. (2024). Attention vs LSTM: Improving word-level BISINDO recognition. arXiv preprint arXiv:2409.01975.