

Real-time recognition of Indonesian sign language using recurrent neural network

Yoel Andreas ¹, Suci Dwijayanti ¹, Hera Hikmarika ¹, Bhakti Yudho Suprpto ¹

¹ Department of Electrical Engineering, Universitas Sriwijaya, Jl. Raya Palembang Prabumulih KM 32, Indralaya 30662, Indonesia

ARTICLE INFO

Article history:

Received December 1, 2023

Revised December 19, 2023

Published February 1, 2024

Keywords:

BISINDO;
Recurrent neural network;
Realtime;
Feature;
Optimizer;
Sign

ABSTRACT

Hand gestures serve as a vital means of communication for deaf individuals. They often face communication challenges in their daily interactions due to the language barrier. This underscores the necessity of sign language interpreters. However, prevailing methods primarily rely on the Indonesian Sign Language System (SIBI), despite the widespread use of Indonesian Sign Language (BISINDO) for communication. Additionally, the effectiveness of these methods hinges greatly on the accuracy of feature extraction. To address this limitation, this study introduces a Recurrent Neural Network (RNN) approach for BISINDO interpretation. Data acquisition involved the use of a webcam to capture video data, subsequently transformed into frames and arrays. Collected from three respondents, the dataset comprises 3,240 videos and 97,200 array data points, encompassing letters and numbers. Among the tested parameters, training results indicate that utilizing the Adam optimizer with a learning rate of 0.0001 and 500 epochs yields the highest accuracy and minimal loss compared to other configurations. Subsequently, this model underwent real-time testing, conducted five times for 36 classes, achieving an accuracy of 81.67%. It is important to note that errors may arise due to similarities within hand signal language, particularly involving characters such as I, J, D, P, M, and N.

This work is licensed under a [Creative Commons Attribution-Share Alike 4.0](https://creativecommons.org/licenses/by-sa/4.0/)



Corresponding Author:

Suci Dwijayanti, Department of Electrical Engineering, Universitas Sriwijaya, Jl. Raya Palembang Prabumulih KM 32, Indralaya 30662, Indonesia
Email: sucidwijayanti@ft.unsri.ac.id

1. INTRODUCTION

Sign language is a unique form of communication that does not rely on the sounds of human speech or written symbols. Deaf individuals typically use sign language as their primary means of communication. According to data from the World Health Organization (WHO) as of March 15, 2018, there were 466 million deaf individuals, including 34 million children, constituting more than 5% of the global population [1].

In everyday life, deaf individuals often face challenges when communicating with those who do not understand sign language. Therefore, there is a growing need for sign language translators to facilitate communication between deaf individuals and those unfamiliar with sign language. One solution is to convert sign language into written text.

Various methods have been implemented to convert hand gestures to text. For instance, [2] utilized the sign language MNIST dataset with a modified Inception V3 model of a convolutional neural network (CNN). Meanwhile, [3] developed continuous sign language using a recurrent neural network as the sequence learning module on multimodal data. Another approach involves using an adapted deep convolutional neural network with the Hand Gesture Dataset LSP [4]. Video-based sign language has been proposed using a hierarchical attention network with latent space [5]. In another study, a lightweight model based on You Only Look Once

(YOLO) v3 has been developed without any further preprocessing [6]. Yet another study also utilized CNN for real-time dynamic hand gestures [7].

The methods employed in converting hand gestures into text across various sign languages exhibit weaknesses as they heavily rely on input accuracy and hand positioning to generate accurate outputs. In addition, the Indonesia Sign language is still limited. Some studies only focus on Sistem Isyarat Bahasa Indonesia (SIBI) [8][9][10] whereas most of the deaf prefer to use BISINDO [11] instead of SIBI. Given the shortcomings identified in previous studies, this research aims to develop a hand gesture-to-text conversion system for BISINDO using the Recurrent Neural Network (RNN) method. The RNN approach is anticipated to address issues related to input accuracy, specifically in the domain of feature extraction, while also facilitating real-time implementation. Previous work by Heyuan Guo, Yang Yang, and Hua Cai [12] employed RNN to convert diverse gestures into text, albeit with the use of Kinect for hand movement detection. In contrast, this research utilizes real-time camera input.

The structure of this paper is as follows: Section 2 provides a brief explanation of RNN. Then, Section 4 outlines the method, followed by results and discussion. Finally, the conclusion is summarized in the last section.

2. RECURRENT NEURAL NETWORK

Recurrent Neural Network (RNN) is a form of Artificial Neural Networks (ANN) architecture specifically designed to process sequential data. RNN does not discard information from the past in its learning process. This is what distinguishes RNNs and ordinary ANNs. RNN is able to store memories that allow it to recognize data patterns well, then use them to make accurate predictions.

In each process, the output is not only a function of that sample, but also based on the internal state that is the result of processing previous samples. The way that RNNs are able to store information from previous samples is by looping within their architecture, which automatically keeps information from the past stored. The general architecture of an RNN can be seen in Fig. 1.

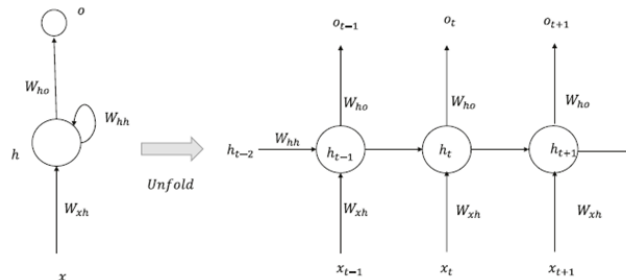


Fig. 1. Architecture RNN

In theory, an RNN can process sequential data of any length. The amount of previous data that becomes input feedback for the next stage is as long as the input pattern to be recognized. However, in practice, the recurrent topology is expressed by cloning as many RNN units as the number of backward steps required. These rows of clones will form a long string of neural networks. Because the training data used is backpropagation like neural networks in general, strands that are too long will tend to experience vanishing gradient problems which cause the gradient value to become very small or very large when the input sequence is too long. The equations that can be used to find the hidden state and output values are shown in (1) and (2).

$$h_t = f((h_{t-1}W_{hh}) + (h_tW_{xh})) \quad (1)$$

$$o_t = f(h_tW_{ho}) \quad (2)$$

where x_t is the input at time step, W_{xh} is the weight that converts input x_t to state h_t , W_{hh} is the weight that converts the previous state (h_{t-1}) to the current state (h_t), o_t is the output at time step, W_{ho} is the weight that maps the calculated state to the output, and f is an activation function such as tanH, sigmoid, or ReLU.

In training using RNN, it is difficult to do because of the vanishing/exploding gradient problem. To overcome this, a variant of RNN called LSTM was introduced [13]. Long Short-Term Memory (LSTM) is a variant of RNN that can overcome the exploding/vanishing gradient problem that can cause inaccurate accuracy in training a model. Fig. 2. shows the structure of the LSTM.

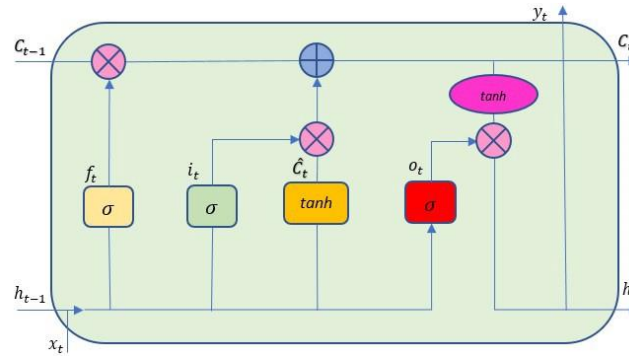


Fig. 2. Structure LSTM

In LSTM, there are three gates that distinguish between LSTM and regular vanilla RNN, namely forget gate, input gate, and output gate. In the forget gate section, information that is not needed will be discarded using a sigmoid function so as not to cause exploding/vanishing gradient. The equation that can be used to show the value of the forget gate can be seen in (3). In the input gate section, information will also be selected to be updated to the cell state using the sigmoid activation function. The equation that can be used to show the value of the input gate can be seen in equation 4. Then in the output gate section, the sigmoid function will be run again to get the output value in the hidden state and place the cell state in the tanh activation function. Equations that can be used to show the value of the output gate, cell state, and hidden state can be seen in equations 5, 6, 7.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$c_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (6)$$

$$h_t = o_t * \tanh(c_t) \quad (7)$$

3. METHODS

The method used for sign language recognition will be implemented using a Recurrent Neural Network (RNN) programmed in Python software. Videos will be captured using a Logitech C922 camera, and each frame will be captured at a rate of 10 frames per second (FPS). The process involved in hand gesture-to-text conversion can be observed in Fig. 4.

Testing is conducted by calculating the success rate of hand gesture-to-text conversion. This test employs the confusion matrix method, which compares the system's classification results with the actual classification results. You can find the confusion matrix in Table 1.

Table 1. Confusion Metric

Predicted Value	Actual Value	
	True Positive (TP) False Negative (FN)	False Positive (FP) True Negative (TN)

Table 1 contains both predicted values, provided by the RNN system, and actual values that have been determined. The accuracy value can be calculated using Equation 8.

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (8)$$

True Positive (TP) represents correctly identified positive data by the system, while False Positive (FP) denotes incorrectly identified positive data. True Negative (TN) stands for correctly identified negative data, while False Negative (FN) indicates incorrectly identified negative data.

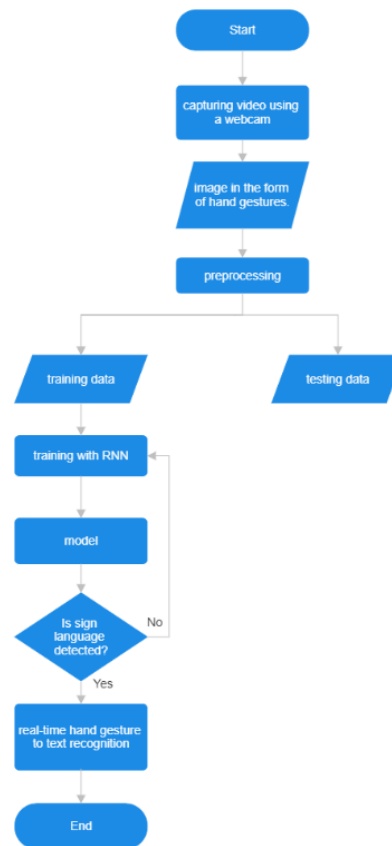


Fig. 4. System design flowchart

4. RESULTS AND DISCUSSION

4.1. Data Collection Process

This research begins with the data collection phase, involving the recording of videos that are later processed by MP Holistic to create arrays. These arrays then undergo training using the Recurrent Neural Network (RNN) method.

For this study, a Logitech C922 webcam camera is used to collect the video dataset. During the data collection process, the respondents are positioned 60 cm away from the camera, placed in the center, and facing it. The data collection room is well-lit to ensure the camera captures the necessary letter and number characters.

Three Electrical Engineering students from the Faculty of Engineering at Universitas Sriwijaya participated in this data collection. The study incorporates 36 classes, comprising 26 letters of the alphabet (A to Z) and 10 numbers (1 to 10). Each class consists of 90 videos, each containing 30 frames, which are converted into arrays. This results in a total of 97,200 npy data files. Thirty video samples are collected for each class. You can view examples of the collected data in Figs. 5. and 6.



Fig. 5. Image of the letter “F”

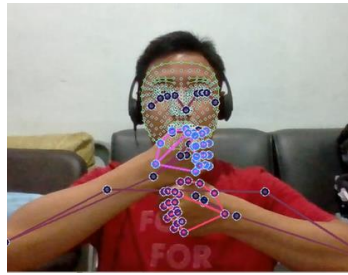


Fig. 6. Image of the letter “G”

4.2. Dataset Preprocessing Stage

Before the data can be utilized in RNN training, it undergoes a preprocessing phase to simplify the training process. This preprocessing consists of two steps: converting video data into frames and transforming each frame into an array.

Prior to data processing, the acquired videos are initially converted into frames. From the 90 videos per class that were captured, 30 frames are selected from each video, resulting in a total of 97,200 frames.

The dataset, now in the form of frames, needs further conversion into arrays. This step is necessary because feature extraction is performed using MP Holistic, resulting in array data containing hand positions for each frame. The array data will be stored in .npy format.

4.3. RNN Training

Before commencing the training process, it is essential to determine the RNN architecture to be utilized. In this study, several tests were conducted to identify the architecture with the most optimal parameters.

For this research, the hardware specifications employed during training include an NVIDIA GEFORCE 1650 graphics card, Intel i7 processor, and 8 GB of RAM. The hardware that supports the training process includes NVIDIA GPU drivers, Tensorflow 2.31, Python 3.7, and other necessary components. Each trained model will be saved in .h5 format.

During the training process using the RNN method, the optimizer employed includes the Adam optimizer and the Stochastic Gradient Descent (SGD) optimizer.

During the RNN training process, achieving the best model is an iterative process involving trial and error. The chosen model should exhibit both the highest accuracy value and the lowest loss value. This stage entails modifying various parameters within the RNN, including the optimizer, learning rate, and the number of epochs.

To begin the RNN training process, the first step is to determine the most optimal optimizer. In this study, two training sessions were conducted using both the Adam optimizer and the Stochastic Gradient Descent (SGD) optimizer. The comparison of the results obtained with these two optimizers is presented in Fig. 7.



Fig. 7. Comparison loss optimizer Adam and SGD

From the figure, it is evident that the Adam optimizer is more effective in minimizing the loss value. Consequently, the Adam optimizer is employed in this research.

The value of the learning rate can also significantly impact the model's loss value and accuracy. In this study, learning rate parameters of 0.01, 0.001, and 0.0001 are used. Table 2 displays the effects of the learning rate on both the loss value and accuracy.

Table 2. Loss and Accuracy from The Effect of Learning Rate (300 Epochs)

No	Learning Rate	Optimizer	Loss	Accuracy
1	0.0001	Adam	0.06758	0.976
2	0.001	Adam	1.252	0.4425
3	0.01	Adam	3.587	0.02339

From Table 2, it is evident that a learning rate of 0.0001 yields the smallest loss value and the highest accuracy. Consequently, in this study, the optimal learning rate utilized is 0.0001.

In the RNN training process, the number of epochs also has a significant impact on accuracy and loss values. To determine the optimal number of epochs, this research conducted training for both 300 and 500 epochs, while maintaining consistent parameters, including a learning rate of 0.0001, 7 hidden layers, a dropout rate of 0.2, and the Adam optimizer. Table 3 illustrates the effects of the number of epochs on both loss value and accuracy.

Table 3. Loss and Accuracy Results from The Effect of Epoch (Learning Rate 0.0001)

No	Epoch	Optimizer	Loss	Accuracy
1	300	Adam	0.06758	0.976
2	500	Adam	0.07016	0.9812

From the figures and tables above, it's evident that the epoch value influences the accuracy. In the upcoming tests to find the optimal model, we will evaluate two models: one with 300 epochs and another with 500 epochs.

Testing aims to evaluate the performance of the RNN LSTM model obtained during the training phase for real-time Indonesian sign language recognition. This testing process was repeated five times for each class, with five different respondents participating. It was conducted on the two best models obtained, which were trained for 300 and 500 epochs, respectively. Both models share the same parameters: a learning rate of 0.0001, three LSTM layers, three Dense layers, a dropout rate of 0.2, and the Adam optimizer.

Following the selection of the best model based on the comparison of the two epochs, real-time testing involved assessing the impact of varying lighting conditions. The results of the real-time tests showed a test accuracy of 66.89% for the 300-epoch model and 81.67% for the 500-epoch model. Table 4 presents the test results for the model trained with 500 epochs.

Table 4. Test Results from Model with 500 Epochs

Class	Accuracy	Error	Class	Accuracy	Error
1	80%	4(20%)	I	20%	J(60%), 1(20%)
2	80%	3(20%)	J	60%	C(20%), 1(20%)
3	40%	L(20%), O(20%), V(20%)	K	100%	-
4	60%	R(20%), V(20%)	L	60%	V(20%), 1(20%)
5	40%	V(60%)	M	100%	-
6	100%	-	N	60%	M(40%)
7	100%	-	O	100%	-
8	100%	-	P	100%	-
9	40%	8(40%), 10(20%)	Q	80%	S(20%)
10	100%	-	R	60%	O(20%), 1(20%)
A	100%	-	S	80%	T(20%)
B	100%	-	T	80%	B(20%)
C	100%	-	U	100%	-
D	60%	A(20%), P(20%)	V	100%	-
E	80%	U(20%)	W	100%	-
F	80%	B(20%)	X	100%	-
G	100%	-	Y	100%	-
H	80%	D(20%)	Z	100%	-

After obtaining the optimal model, it will undergo further testing to assess the model's performance in recognizing sign language under varying room lighting conditions. The selected model for this testing phase utilizes 500 epochs. Two tests were conducted, one in bright lighting and another in dim lighting.

The tests carried out in dim lighting revealed the influence of room lighting on the model's ability to recognize sign language. In darker environments, MP Holistic faces greater difficulty in capturing the hand's shape due to the shadows caused by dim lighting, which can lead to the model's inability to recognize sign language. The test conducted in a dimly lit room yielded a success accuracy of 76.11%.

Errors made by the model in recognizing sign language can be attributed to three factors: character similarity, hand positioning, and room lighting conditions.

The RNN tests demonstrate the system's proficiency in recognizing sign language. However, because these tests are conducted in real-time, certain conditions must be carefully observed. In the tests conducted for this study, the camera was positioned at a distance of 60 cm, the respondent's face was centered and facing the camera, and hand positioning played a crucial role in the testing process.

5. CONCLUSION

Based on the results of the conducted research, it can be concluded that RNN LSTM is a viable tool for recognizing Indonesian Sign Language (BISINDO). The dataset employed in this study comprised 3,240 video samples captured using a Logitech C922 webcam. The findings indicated that the most effective model resulted from training with the Adam optimizer, a learning rate of 0.0001, and 500 epochs. In conclusion, this research demonstrates the capability of RNN in effectively recognizing Indonesian Sign Language. Errors observed in characters I, J, D, P, M, N can be attributed to their similar shapes. The real-time testing yielded an accuracy rate of 81.67%.




For future research, it is anticipated that the application of the recurrent neural network method in Indonesian sign language recognition can extend to recognizing words within the language. Additionally, further research exploring two-way communication, encompassing both hand gestures to speech and speech to hand gestures, is encouraged.

REFERENCES




- [1] WHO Media Center, "Deafness." [Online]. Available: <https://www.who.int/news-room/facts-in-pictures/detail/deafness>. (Nov 10, 2023).
- [2] M. Agrawal, R. Ainapure, S. Agrawal, S. Bhosale, and S. Desai, "Models for Hand Gesture Recognition using Deep Learning," *2020 IEEE 5th Int. Conf. Comput. Commun. Autom. ICCCA 2020*, pp. 589–594, 2020, doi: 10.1109/ICCCA49541.2020.9250846.
- [3] R. Cui, H. Liu, and C. Zhang, "A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training," *IEEE Trans. Multimed.*, vol. 21, no. 7, pp. 1880–1891, 2019, doi: 10.1109/TMM.2018.2889563.
- [4] A. A. Alani, G. Cosma, A. Taherkhani, and T. M. McGinnity, "Hand gesture recognition using an adapted convolutional neural network with data augmentation," *2018 4th Int. Conf. Inf. Manag. ICIM 2018*, pp. 5–12, 2018, doi: 10.1109/INFOMAN.2018.8392660.
- [5] J. Huang, W. Zhou, Q. Zhang, H. Li, and W. Li, "Video-based sign language recognition without temporal segmentation," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 2257–2264, 2018, doi: 10.1609/aaai.v32i1.11903.
- [6] A. Mujahid *et al.*, "Real-time hand gesture recognition based on deep learning YOLOv3 model," *Appl. Sci.*, vol. 11, no. 9, 2021, doi: 10.3390/app11094164.
- [7] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," *Proc. - 14th IEEE Int. Conf. Autom. Face Gesture Recognition, FG 2019*, 2019, doi: 10.1109/FG.2019.8756576.
- [8] D. M. Adimas, E. Rakun, and D. Hardianto, "Recognizing Indonesian Sign Language Gestures Using Features Generated by Elliptical Model Tracking and Angular Projection," *Proc. - 2019 2nd Int. Conf. Intell. Auton. Syst. ICoIAS 2019*, no. March, pp. 25–31, 2019, doi: 10.1109/ICoIAS.2019.00011.
- [9] I. Mahfudi, M. Sarosa, R. Andrie Asmara, and M. Azrino Gustalika, "Indonesian Sign Language Number Recognition using SIFT Algorithm," in *IOP Conference Series: Materials Science and Engineering*, Apr. 2018, vol. 336, no. 1, doi: 10.1088/1757-899X/336/1/012010.
- [10] K. Anggraini, E. Rakun, and L. Y. Stefanus, "Recognizing the Components of Inflectional Word Gestures in Indonesian Sign System known as SIBI (Sistem Isyarat Bahasa Indonesia) by using Lip Motion," *Proc. Int. Conf. Electr. Eng. Informatics*, vol. 2019-July, no. July, pp. 384–389, 2019, doi: 10.1109/ICEEI47359.2019.8988806.
- [11] T. Handhika, I. Sari, R. Ilfesta, M. Zen, and D. P. Lestari, "The Generalized Learning Vector Quantization Model to Recognize Indonesian Sign Language (BISINDO)." In *2018 Third International Conference on Informatics and Computing (ICIC)*, pp. 1–6. IEEE, 2018.
- [12] H. Guo, Y. Yang, and H. Cai, "Exploiting LSTM-RNNs and 3D Skeleton Features for Hand Gesture Recognition," *WRC SARA 2019 - World Robot Conf. Symp. Adv. Robot. Autom. 2019*, pp. 322–327, 2019, doi: 10.1109/WRC-SARA.2019.8931937.
- [13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

BIOGRAPHY OF AUTHORS

Yoel Andreas, was born on March 22, 1999 in Prabumulih (South Sumatra, Indonesia). He was a graduate of the Sriwijaya University of Palembang in the major of Electrical Engineering last year. He is a member of the Computer and Robotic Laboratory in his major. His final project in bachelor's degree was about image processing. He concerns about learning more about image processing and deep learning. Email: yoelakame@gmail.com

Suci Dwijayanti,    received her M.S. in Electrical and Computer Engineering from Oklahoma State University, Stillwater, OK, USA, in 2013. She received a Fulbright scholarship for her Master's degree. In 2018, she received her Doctoral degree from the Graduate School of Natural Science and Technology, Kanazawa University, Japan. Her research interests include signal processing and machine learning. She also worked as an engineer at ConocoPhillips Indonesia Inc. Ltd. from 2007 to 2008. Since 2008, she has been with the Department of Electrical Engineering at Universitas Sriwijaya, Indonesia. She became an IEEE member in 2014. She can be contacted at email: sucidwijayanti@ft.unsri.ac.id. Orchid:

Hera Hikmarika, was born on December 7th 1978 in Palembang (South Sumatra, Indonesia). She is a graduate of the Sriwijaya University of Palembang with a major in electrical engineering. Her master program is in Electrical Engineering of Universitas Gadjah Mada (UGM). She is an academic staff of Electrical Engineering of the Sriwijaya University of Palembang as a lecturer. She concerns with major control and intelligence. Email: herahikmarika@ft.unsri.ac.id

Bhakti Yudho Suprpto,    was born on February 11, 1975 in Palembang (South Sumatra, Indonesia). He is a graduate of Universitas Sriwijaya with a major in Electrical Engineering. His Master's and Doctoral degrees were in Electrical Engineering from Universitas Indonesia (UI). He is an academic staff of Electrical Engineering of Sriwijaya University of Palembang. His research interests are control and intelligent systems. He became an IEEE member in 2014. He can be contacted at email: bhakti@ft.unsri.ac.id