SIBI Dynamic Gesture Translation Using MediaPipe and Long Short-Term Memory in Real-Time

Rivano Ardiyan Taufiq Kurniawan and Wilis Kaswidjanti*

Department of Informatics, Faculty of Industrial Engineering, Universitas Pembangunan Nasional "Veteran" Yogyakarta, Sleman 55281, Indonesia
wilisk@upnyk.ac.id

Abstract. Commonly used image processing classification methods like Artificial Neural Network and Convolutional Neural Network are considered successful for sign language identification. However, they perform well only with static data and face limitations in handling sequential and dynamic data like Indonesian Sign Language System (SIBI) sign gestures. To address this, this research uses the Long Short-Term Memory (LSTM) method which has a flexible architecture and can adjust dynamically to accommodate various input sequence lengths, making it reliable in handling sequential data and allowing it to be implemented in real-time systems. This research uses a primary dataset which directly collected by the author, featuring six classes based on question words: "what," "how," "how much," "where," "why," and "who." The 180 original data are augmented into 3060 (510 for each class) with four variations: rotation, zoom in, zoom out, and brightness and contrast adjustments. Data processing utilizes the MediaPipe framework to extract hand landmarks from each data point, saving them as numerical data in NumPy array format. Thus, instead of detecting the entire image susceptible to background noise, detection focuses solely on landmarks indicating hand and finger positions. With a data split of 2616 for training, 153 for testing, and 291 for validation, the model is constructed with three LSTM layers and three Dense layers. This combination yields a categorical accuracy of 99.85%, a loss of 0.0059, validation categorical accuracy of 100%, and validation loss of 0.0064 after 150 training epochs.

Keywords: sign language recognition, SIBI, LSTM, MediaPipe, real-time.

1 Introduction

Based on data reported from Information System for Persons with Disabilities by Ministry of Social Affairs of the Republic of Indonesia published in research by Data and Information Center of the Ministry of Health of the Republic of Indonesia [1], among all Indonesian people with disabilities, 7.03% of them are deaf and 2.57% are speech impaired. In carrying out their daily activities, these people certainly need a special media or way of communication considering their conditions that do not allow verbal communication.

A. Putro Suryotomo and H. Cahya Rustamaji (eds.), *Proceedings of the 2023 1st International Conference on Advanced Informatics and Intelligent Information Systems (ICAI3S 2023)*,
Advances in Intelligent Systems Research 181,
https://doi.org/10.2991/978-94-6463-366-5_6

In Indonesia itself, there is a standardized sign system that has been standardized by the Indonesian government called the Indonesian Sign Language System (SIBI). However, in practice SIBI is still difficult to understand by ordinary people because it requires special learning and training. In addition, conventional manual interpretation is currently considered impractical due to the limited availability of sign language interpreters [2], especially those who master SIBI. As a result, the effectiveness of SIBI in bridging communication between people with disabilities and the general public is sometimes not optimally achieved. In the end, SIBI is more often used only as a medium of communication between people with disabilities.

To overcome this problem, it is necessary to develop a system that is able to translate SIBI into textual form. With this system, it is hoped that it will facilitate the communication process between deaf people and people who are unfamiliar with sign language. There are various classification methods that can be used to identify sign language and translate it into text form, some of which are Artificial Neural Network [3], Convolutional Neural Network [4] [5] [6] [7] [8], and Long Short-Term Memory [9] [10] [11] [12] [13].

Artificial Neural Network (ANN) has advantages in distributed memory, has the ability to tolerate errors, and is able to work with limited knowledge. However, ANN has a weakness in the difficulty of determining the right network architecture and the lack of ability to handle sequential data [3].

Convolutional Neural Network (CNN) has the advantages of having a wide choice of architectures [6] [7] [8], can produce high accuracy in image recognition problems in a fast time [5], and is able to process long sequential data such as video [5] [6] [7]. Moreover, CNN is also one of the most popular methods used in the case of sign language translation. However, CNN has the drawback of requiring a large amount of training data and the possibility of poor performance on small datasets [4]. In addition, the architecture that needs to be predefined leads to a lack of flexibility that CNNs have. So, to be used to handle sequential data with high accuracy, modifications are needed with the addition of pre-processing stages and combination with other classification methods [4] [5] [6] [7] [8].

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network that can solve the problems of ANN and CNN. Unlike ANN and CNN that require a predefined architecture, LSTM has a flexible architecture and can adjust dynamically to accommodate various input sequence lengths [11] [13]. This also makes LSTM more reliable in handling sequential data [10] [11] [12] [13]. In addition, LSTM also requires less data compared to ANN and CNN to achieve good results due to its ability to retain and update information selectively over time [10] [13]. Obviously, the capabilities of the LSTM method are very appropriate to be implemented in the SIBI translation system which has dynamic gestures and allows it to be implemented in a system that runs in real-time. The use of LSTM method in sign language translation has also been proven in previous studies with good results [9] [10] [11] [12] [13].

In addition to the machine learning method used, the model built will be greatly influenced by the quality of the data used. In [3] and [11] the data was collected using the Motion Leap Sensor tool, while in [6] the data was collected using the Microsoft Kinetic tool. These tools can greatly help the gesture classification process by the

model, because instead of detecting the entire image which is prone to background noise, the detection focus is only on landmarks or dots and lines that indicate the position of hands and fingers. However, the downside of using a physical device is that the entire data collection and testing process must also be done using the device. This certainly makes the data collection process very inefficient. Therefore, this research will utilize a technology from Google called MediaPipe that is capable of reading landmarks from a video. This technology was also used in [10] and was able to significantly affect the accuracy of the model as the processed data showed clearer and more precise information. However, the cue data used in that study was mostly static and simple.

In overcoming the problem of handling sequential and dynamic gesture data, this research will combine the LSTM method with MediaPipe technology. MediaPipe will be applied in the data capture process, which is then processed with the model built using the LSTM method. The result is a system that is able to automatically translate gestures captured by the camera into text in real-time.

2 Method and Design

The research uses an experimental quantitative method with stages including literature study, data collection, data pre-processing, model building, model training, and ends with analysis and evaluation of results.

2.1 Literature Study

The research began by conducting a literature study of previous research, namely tracing library sources such as books, scientific articles, journals, papers, and others. This stage aims to find information relevant to the Indonesian Sign Language System (SIBI), Sign Language Recognition (SLR), hand gesture detection, and the application of image processing methods. Thus, it is expected that the best solution can be found based on research that has been done before.

2.2 Data Collection

After the solution is found, the stage continues by collecting SIBI gesture image data. The data is obtained from the direct capture process by the author using the NYK Nemesis A95 Albatross 2K QHD Webcam camera (4MP, 30FPS). The video data obtained is then split into a series of frames consisting of 30 images. The total original data used is 180 data and divided into six classes, namely "what", "how", "how many", "where", "why", and "who" with 30 data each. Using the Holistic module of the MediaPipe framework, keypoints from each hand landmark in the data are then detected, extracted, and stored as a NumPy array.

2.3 Data Pre-processing

The data that has been collected is still in the form of raw data, so it needs to be prepared first into a more structured and higher quality format for further analysis. The pre-processing includes several steps, namely: data augmentation, augmentation data processing, data labeling, and data splitting. Data augmentation is performed with four variants shown in the following table:

Table 1. Data Augmentation

No	Parameter	Value	Description
1	Rotation	15, 30, 45, 315, 330, 345	Image rotation is done six (6) times with rotation degrees of 15, 30, 45, 315, 330, and 345 degrees.
2	Zoom	0.7, 0.8, 0.9	Enlargement of the image size is done three (3) times by cutting the image so that it leaves 0.7, 0.8, and 0.9 percent of the original size.
3	Zoom Out	0.7, 0.8, 0.9	Shrinking the image size is done three (3) times by adding a border around the image so that the image size becomes 0.7, 0.8, and 0.9 percent smaller than the original size.
4	Brightness and Contrast	0.5, 1.0, 1.5, 2.0	Changes in image brightness and contrast are made with an interval of 0.5. The resulting image has brightness and contrast values of 0.5x, 1.0x, 1.5x, and 2.0x from the original image.

From the data augmentation performed, one (1) sequence will be multiplied sixteen (16) times, so that for each class or sign language vocabulary used will have as much as 510 data and the total data used is 3060 data. Just like the original data, the augmented data obtained will also have its keypoints extracted and stored in the form of a NumPy array. The NumPy array data containing the keypoints extraction results of the original image and the augmented image will be used as the data set in this research.

Next, the data labeling process is carried out to categorize the data according to the predetermined sign language vocabulary classes, namely "what", "how", "how much", "where", "why", and "who". This stage ends with splitting the data to divide the data into training data and testing data with a ratio of 95:5, so that the total training data is 2907 and testing data is 153. The data splitting ratio used in this case refers to research [10] [14].

2.4 Model Building

The machine learning model is built using the Long Short-Term Memory method and uses the NumPy data array generated from the extraction process that has been carried

out previously. The model has a total of 6 (six) layers, consisting of three LSTM layers and three Dense layers. The first LSTM layer has 64 units with input shape values of 30 (number of frames for each sequence) times 126 (number of features generated from keypoint detection on both hands for each x, y, and z point). Then, in the second LSTM layer, the number of units is increased to 128 to allow for a higher level of representation and complexity in the learned features. Then, in the third LSTM layer, the number of units is reduced back to 64 to maintain a balance between the capacity and complexity of the model.

The features that have been generated from the third LSTM output sequence are then processed in the first Dense layer with 64 units, followed by the second Dense layer with 32 units, and ended with the third Dense layer with 6 units (corresponding to the number of classes). In each Dense layer, the number of units used decreases over time, this is done to reduce the dimension of the learned representation so that it can extract features with a higher level.

The activation function used in the first LSTM layer up to the second Dense layer is ReLU (Rectified Linear Unit), while for the third Dense layer (output layer) is Softmax, where it serves to generate the probability distribution of the input features against each class.

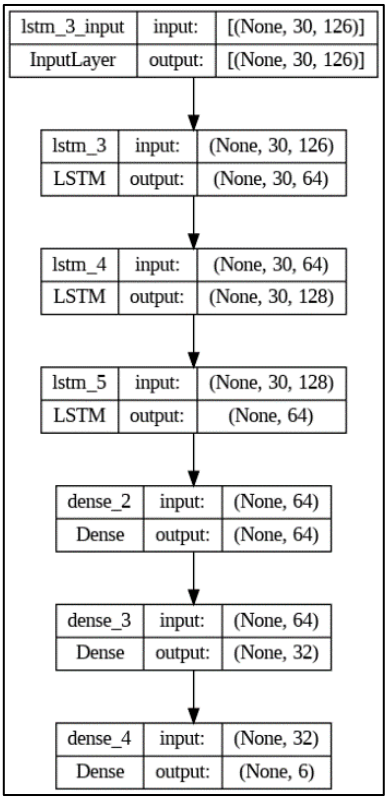

Fig. 1. Shape Details (Dimensions) of Data in the Model

2.5 Model Training

The model is trained through two stages, namely: data compiling and data training. The data compiling step is done with the Adaptive Moment Estimation (Adam) optimization method, Categorical Crossentropy loss function, and Categorical Accuracy metrics. The selection of loss functions and metrics refers to research [9] and [14].

In the training data stage, the model is made to run with an epoch value of 150 until the accuracy value approaches 1 and the training loss approaches 0. 10% of the total training data (291 data) will be used as validation data. The rest, 90% of the training data (2616 data), will be used to train the model. In training the data, the batch size value used is the default value, which is 32 with a total of 82 batches available to train all data in this model.

2.6 Analysis and Evaluation of Results

In the final stage, data plotting will first be carried out, where a comparison graph of the loss trend and accuracy obtained between training data and validation data will be displayed. This graph is used to visualize the performance of the model during training by showing the trend of loss and accuracy from epoch to epoch, which can help in evaluating the learning progress of the model, detecting overfitting or underfitting, tuning the model parameters, and providing a visual representation of the training process.

In addition, it will also be analyzed whether the results obtained successfully answer the problems that have been formulated previously. The model will also be evaluated by checking its accuracy through testing using confusion matrix and real-time system testing. The things obtained at this stage will be used as a reference in drawing research conclusions.

3 Results and Discussion

3.1 Results

From the model training conducted with an epoch value of 150, the loss value is 0.0059, categorical accuracy is 0.9985, validation loss is 0.0064, and validation categorical accuracy is 1.

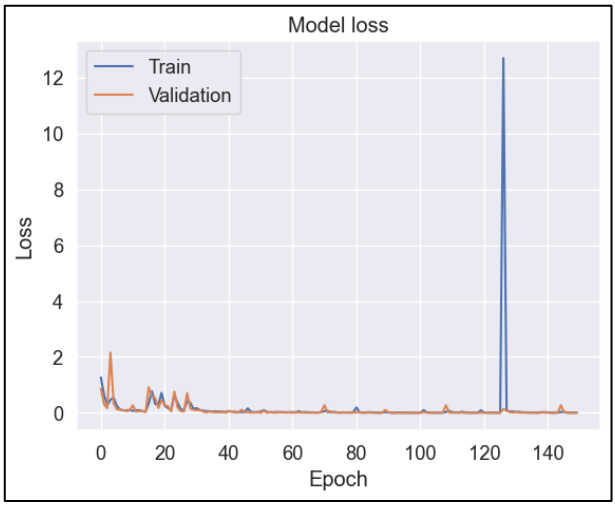

Fig. 2. Model Loss Graph

The model loss graph in Fig. 2 shows the changes in the loss value of train and validation data that occur in the model for each epoch. It can be seen that the loss value tends to fluctuate at epoch values between 1-32, then becomes relatively stable at the epochs afterwards. The exception is at the 124th epoch value where there is an extreme increase in the loss value of the train data.

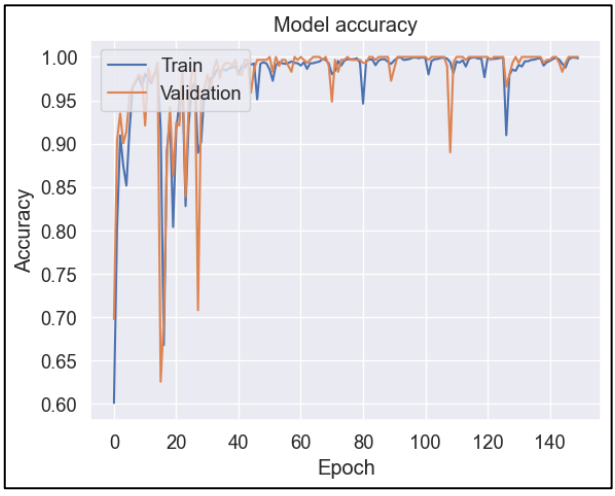

Fig. 3. Model Accuracy Graph

The accuracy model graph in Fig. 3 shows the changes in the accuracy value of train and validation data that occur in the model for each epoch. Almost the same as the loss graph, the accuracy value tends to experience quite fluctuations in the epoch value

between 1-28, then becomes relatively stable in the epochs afterwards although there are still some fluctuations that are still at normal levels at certain epoch points.

From these two graphs, it can be interpreted that the model is able to perform very well on the given data and does not experience overfitting or underfitting. To prove this, a confusion matrix is created to compare the model's prediction results to the actual values. From a total of 153 testing data, 28 data were taken from the "what" class, 28 data from the "how" class, 23 data from the "how much" class, 19 data from the "where" class, 27 data from the "why" class, and 28 data from the "who" class. As a result, the entire testing data is perfectly predicted by the model. The confusion matrix output is shown in Table 2.

Table 2. Confusion Matrix

	Predicted						
Actual	Class	*apa* (what)	*bagaimana* (how)	*berapa* (how much)	*di mana* (where)	*men-gapa* (why)	*siapa* (who)
	apa (what)	28	0	0	0	0	0
	bagaimana (how)	0	28	0	0	0	0
	berapa (how much)	0	0	23	0	0	0
	di mana (where)	0	0	0	19	0	0
	mengapa (why)	0	0	0	0	27	0
	siapa (who)	0	0	0	0	0	28

The average accuracy value of 1.0, precision of 1.0, recall of 1.0, and F1-score of 1.0 were obtained. A summary of the overall model performance results is shown in Table 3, where the support column shows the amount of data, the macro avg row shows the average metric value regardless of the size and balance of each class, and the weighted avg row shows the average metric value considering the size and balance of each class.

Table 3. Model Performance Results

	Precision	Recall	F1-Score	Support
apa (what)	1.00	1.00	1.00	28
bagaimana (how)	1.00	1.00	1.00	28
berapa (how much)	1.00	1.00	1.00	23
di mana (where)	1.00	1.00	1.00	19
mengapa (why)	1.00	1.00	1.00	27
siapa (who)	1.00	1.00	1.00	28
accuracy			1.00	153
macro avg	1.00	1.00	1.00	153
weighted avg	1.00	1.00	1.00	153

Then, to verify that the system built has behaved as expected, system testing is carried out using the black box testing method. The system that runs in real-time is tested by running it directly to observe the functionality and behavior of the system when given input as well as producing output correctly. From the tests that have been carried out, the system has fulfilled all existing test scenarios, namely:

1. The user can see the results of the camera capture along with the sign of his hand position.
2. The user can see a bar that shows the percentage of gestures performed against the predetermined classes.
3. The user can see the classification results of SIBI gestures that have been performed in textual form.

To provide a clearer picture, the following figure is the result of system testing when given gesture input from one of the classes used.

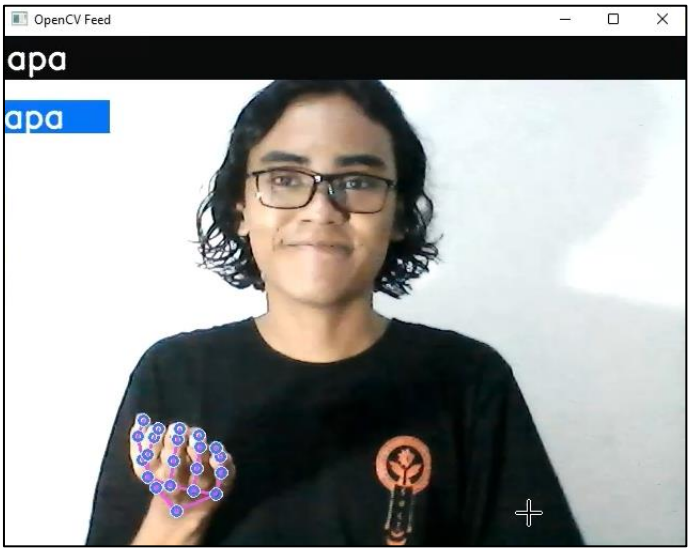

Fig. 4. Real-Time Testing Results for *apa* (what) Class

Fig. 4 shows the display of the system when given the input of the gesture "what". The system successfully classifies the gesture correctly, as indicated by the output of a blue bar representing the "what" class and the display of the word "what" in the black background area at the top of the screen.

3.2 Discussion

The excellent model testing results prove that the combination of the model built using the Long Short-Term Memory method with data processing using the MediaPipe framework is successful in overcoming the problem of handling sequential and dynamic SIBI gesture data. However, the success of real-time testing can be affected by several conditions, such as lighting, background noise, and the wholeness of the hands captured by the camera during the gesture demonstration.

4 Conclusions and Suggestions

The combination of the Long Short-Term Memory model and the MediaPipe framework is able to overcome the problem of handling sequential and dynamic SIBI gesture data through the translation of SIBI gestures into textual form in real-time. This is evidenced by the results of the model training process with 150 epochs resulting in a categorical accuracy value of 99.85% and a loss of 0.0059. In addition, testing performance metrics with 5% of the dataset also shows perfect results by producing an accuracy value of 100%, precision of 100%, recall of 100%, and F1-score of 100%.

This research has the potential to be expanded or developed by adding gesture classes that can be classified so that the system can be used to form sentences from gesture

translation results, adjusting the number of layers and units used with the size and complexity of the data used, adjusting the data splitting ratio, epoch value, and batch size to get a more optimal value when training the model, modifying the data augmentation stage to avoid classification confusion due to image truncation and so that the system is able to recognize SIBI gestures in a variety of different conditions, as well as building a more interactive interface and deploying it to a more suitable platform so that the system can achieve its maximum benefits.

References

1. Kemenkes RI, *Infodatin: Disabilitas Rungu*. Jakarta Selatan: Pusat Data dan Informasi Kementrian Kesehatan RI, 2019.
2. A. Venugopalan and R. Reghunadhan, "Applying Deep Neural Networks for The Automatic Recognition of Sign Language Words: A Communication Aid to Deaf Agriculturists," *Expert Syst. Appl.*, vol. 185, no. September 2020, p. 115601, 2021, doi: 10.1016/j.eswa.2021.115601.
3. M. I. Rusydi, Syafii, R. Hadelina, E. Kimin, A. W. Setiawan, and A. Rusydi, "Recognition of Sign Language Hand Gestures Using Leap Motion Sensor Based on Threshold and Ann Models," *Bull. Electr. Eng. Informatics*, vol. 9, no. 2, pp. 473–483, 2020, doi: 10.11591/eei.v9i2.1194.
4. M. E. M. Cayamcela and W. Lim, "Fine-tuning a Pre-trained Convolutional Neural Network Model to Translate American Sign Language in Real-time," *2019 Int. Conf. Comput. Netw. Commun. ICNC 2019*, pp. 100–104, 2019, doi: 10.1109/ICCNC.2019.8685536.
5. S. Shahriar *et al.*, "Real-Time American Sign Language Recognition Using Skin Segmentation and Image Category Classification with Convolutional Neural Network and Deep Learning," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2018-Octob, no. Oktober, pp. 1168–1171, 2019, doi: 10.1109/TENCON.2018.8650524.
6. Suharjito, N. Thiracitta, and H. Gunawan, "SIBI Sign Language Recognition Using Convolutional Neural Network Combined with Transfer Learning and non-trainable Parameters," *Procedia Comput. Sci.*, vol. 179, no. 2019, pp. 72–80, 2021, doi: 10.1016/j.procs.2020.12.011.
7. S. Zhang and Q. Zhang, "Sign Language Recognition Based on Global-local Attention," *J. Vis. Commun. Image Represent.*, vol. 80, no. Agustus, p. 103280, 2021, doi: 10.1016/j.jvcir.2021.103280.
8. W. Tao, M. C. Leu, and Z. Yin, "American Sign Language Alphabet Recognition Using Convolutional Neural Networks With Multiview Augmentation and Inference Fusion," *Eng. Appl. Artif. Intell.*, vol. 76, no. September, pp. 202–213, 2018, doi: 10.1016/j.engappai.2018.09.006.
9. C. K. M. Lee, K. K. H. Ng, C. H. Chen, H. C. W. Lau, S. Y. Chung, and T. Tsoi, "American Sign Language Recognition and Training Method With Recurrent Neural Network," *Expert Syst. Appl.*, vol. 167, no. Desember 2020, p. 114403, 2021, doi: 10.1016/j.eswa.2020.114403.
10. B. Sundar and T. Bagyammal, "American Sign Language Recognition for Alphabets Using MediaPipe and LSTM," *Procedia Comput. Sci.*, vol. 215, pp. 642–651, 2022, doi: 10.1016/j.procs.2022.12.066.
11. A. Mittal, P. Kumar, P. P. Roy, R. Balasubramanian, and B. B. Chaudhuri, "A Modified LSTM Model for Continuous Sign Language Recognition Using Leap Motion," *IEEE Sens. J.*, vol. 19, no. 16, pp. 7056–7063, 2019, doi: 10.1109/JSEN.2019.2909837.

12. D. Guo, W. Zhou, H. Li, and M. Wang, “Hierarchical LSTM for Sign Language Translation,” *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 6845–6852, 2018, doi: 10.1609/aaai.v32i1.12235.
13. T. Liu, W. Zhou, and H. Li, “Sign Language Recognition With Long Short-Term Memory,” *2016 IEEE Int. Conf. Image Process.*, pp. 2871–2875, 2016.
14. A. N. Aliyah, “Implementasi Metode Human Activity Recognition (HAR) Menggunakan Mediapipe Holistics dan Algoritma Long Short Term Memory (LSTM) untuk Menerjemahkan Gerakan Bahasa Isyarat Menjadi Kosa Kata,” Universitas Islam Negeri Syarif Hidayatullah Jakarta, 2022.