Journal of Informatics and Web Engineering

Vol. 4 No. 3 (October 2025) eISSN: 2821-370X

Indonesian Language Sign Detection using Mediapipe with Long Short-Term Memory (LSTM) Algorithm

Wargijono Utomo[1*], Yogasetya Suhanda[2], Harun Ar-Rasyid[3], Andy Dharmalau[4]

[1] Department of Information System, University of Krisnadwipayana, Kota Bekasi, 17411 West Java, Indonesia

[2,3,4] Faculty of Technology, Swadharma Institute of Technology and Business, Jalan Malaka, 11230 West Jakarta, Indonesia

**corresponding author: (wargiono@unkris.ac.id; ORCiD: 0000-0001-9076-1070)*

Abstract - People with disabilities mostly communicate using sign language, but the public still has little understanding of the Indonesian Sign Language System (ISLS). This causes obstacles in daily interactions. Advances in artificial intelligence technology, especially artificial neural networks, open opportunities in sign language recognition, but are still in the development stage. This study aims to build a ISLS sign language recognition model using the LSTM approach and MediaPipe Hands. The method of collecting hand keypoint data, 25 sequences per gesture, and 36 alphabetic and numeric gestures. The dataset is divided into three categories, namely 80% training, 10% validation, and 10% testing. The model developed to handle sequential data from hand gestures using the LSTM architecture. The results of the study can be shown model accuracy of 97.1%, average macro precision of 97%, recall of 96.6%, and F1-score of 96.4% and weighted average precision of 97.4%, recall of 97.1%, and F1-score of 97%. The results show that the combination of LSTM and MediaPipe can detect ISLS gestures with high accuracy. This can be used as a potential solution in automatic sign language translation, so that this model can improve the inclusiveness of communication for people with disabilities. Further research can be developed using a more accurate hand recognition framework, as well as improving data pre-processing, and exploring deep learning (DL) methods such as SSD, YOLO, or Faster-RCNN. In addition, pose and facial recognition can be added to improve accuracy in gesture recognition more comprehensively.

Keywords— Sign Language, Indonesian Sign Language System, Long Short-Term Memory, Mediapipe, Hand Recognition

Received: 15 February 2025; Accepted: 19 June 2025; Published: 16 October 2025

1. INTRODUCTION

Sign language is the primary means of communication for people with disabilities, especially those who have difficulty hearing and speaking [1]. In Indonesia, there are two commonly used sign language systems, namely the ISLS and Indonesian Sign Language (ISL) [2]. According to Republic of Indonesia Ministry of Education and Culture Regulation Number 0161/U/1994, ISLS has been inaugurated as a sign system used in the Special School education curriculum. However, the limited teaching of sign language in public schools has resulted in low public understanding of this sign system. According to data from the 2015 Inter-Census Population Survey (ICPS) issued by the Central

Statistics Agency (CSA), the Indonesian population with hearing difficulties reached 3.35%, while those with speaking difficulties were 1.52% of the total population. This situation demonstrates that there are still major barriers to communication between the public and those with disabilities. Artificial intelligence (AI) technology can be used to automate sign language translation to solve this issue and make communication more inclusive for individuals with impairments.

Artificial intelligence (AI) has found applications in numerous areas but remains in the developmental phase for use in sign language recognition [3], [4], [5]. Other researchers have applied ANN for hand gesture recognition, but this methodology has a weakness in dealing with temporal or sequential data [6], [7]. Considering that sign language includes dynamic hand movements and not just static positions, this approach needs to be able to track and capture change patterns in real time. One type of ANN called recurrent neural networks (RNN) is designed to deal with sequential data, however, it suffers from gradient vanishing which limits the model in learning long sequences [8], [9], [10], [11]. LSTM networks, developed for this purpose, have been shown to perform well in sequential data tasks such as natural language processing (NLP) and gesture recognition [12], [13], [14]. Moreover, the model selection along with the system accuracy and the sign language recognition system can heavily rely on the feature extraction method employed. As one of the more recent developments, MediaPipe can detect and retrieve pertinent features from the face, hands, and body in real items [15], [16], [17]. In conjunction with LSTM, MediaPipe can serve as a promising approach to answer.

Prior studies have created a sign language recognition system using different approaches. As an example, the Convolutional Neural Network (CNN) branch of research focusing on hand shape recognition in sign language achieved remarkable results but struggled with dynamic motion[18], [19]. Other studies with Support Vector Machine (SVM) classification of hand gestures achieved some level of accuracy albeit with significant limitations in optimally capturing sequences of movements [20]. On the other hand, research using LSTM in combination with motion sensor had higher accuracy levels when recognizing dynamic gestures [21], [22], [23]. This approach, however, adds extra hardware which including accelerometers and gyroscopes which makes it impractical for daily use. This means that there is a need to develop a vision-based sign language recognition system that works with images and videos without requiring additional devices.

The goal of this research is to create a SIBI sign language recognition model that combines MediaPipe and LSTM to more precisely identify hand motion patterns. The main contributions in this study include: (i) Development of an LSTM-based system in handling sequential data with sign language. (ii) Use of MediaPipe as a feature extraction method in detecting key hand points in real time. (iii) Evaluation of model performance in recognizing SIBI gestures based on evaluation parameters such as F1-score, recall, accuracy, and precision. The structure of this article includes part 2 literature review, part 3 methods used LSTM model architecture and feature extraction techniques with MediaPipe. Section 4 displays the model performance analysis and experimental data. Section 5 includes conclusions and recommendations for further study.

2. LITERATURE REVIEW

Sign language is the primary means of communication for people with speech and hearing difficulties. ISLS and ISL are the two main sign language systems in Indonesia. ISLS is a system that is formalized by the government and used in formal education, while ISL developed naturally in the deaf community without any standard [24], [25], [26]. Automatic sign language translation faces various challenges, especially in terms of the complexity of hand gestures, facial expressions, and differences in dialects between regions [27], [28], [29], [30]. In addition, the limited availability of sign language datasets is an obstacle to the development of accurate AI-based models [31]. Several studies have developed artificial intelligence models to understand sign language. Computer Vision (CV) and DL-based approaches have become the focus in sign language recognition. Research in [32], [33] uses CNN to recognize hand gestures in static sign language. CNN has achieved high accuracy in static gesture classification, but this method is less effective in handling dynamic gestures, so recent research focuses more on RNN-based models and their derivatives such as LSTM which are more suitable for handling sequential data [34], [35]. Therefore, several other studies have integrated Transformer, which was previously extensively utilized in NLP, in improving the performance of gesture sequence-based sign language recognition [36].

In a sign language recognition system, MediaPipe is one of the frameworks that can be used to detect key points of hands, faces, and bodies in videos in real-time [17]. In using the Hand Tracking API, MediaPipe can capture finger positions precisely without the need for additional sensors. Studies in [37], [38] can be shown that MediaPipe has advantages and processing speed in computational efficiency compared to other methods such as OpenPose. Therefore, MediaPipe is a lighter solution for mobile devices and web-based applications in real-time sign language translation. Therefore, in dealing with sequential sign language problems, a combination approach of MediaPipe and LSTM is needed which has begun to be developed from several recent studies. Studies from [34], [39], the LSTM model can be used to classify hand movement sequences extracted using MediaPipe. The experimental results in this study show that this approach can achieve an accuracy of more than 90% in recognizing hand movements both statically and dynamically. Another study from [40] proposed a combination of MediaPipe-LSTM with Attention Mechanism, allowing the model to focus more on important features in gestures. The results can show significant accuracy improvements compared to traditional methods based only on CNN or RNN.

Previous studies on sign language detection have used traditional CV methods such as HOG and Optical Flow, combined with classification algorithms such as SVM or Random Forest [41]. These techniques apply effectively sign language under specific conditions, but struggle with complex sequential gestures. Recently, there has been a growing adoption of hand-works systems with more advanced technology, particularly CNN, which attempt to recognize hand forms within photographs [42]. Nonetheless, capturing the time-dependent succession of movements is not possible with just a CNN.

To address this problem, numerous researches have applied RNN, particularly the LSTM architecture, due to its advantages in sequential data analysis [43]. However, these techniques often make use of unmapped images, or skeletal data, derived from tedious and complex steps. This study proposes a different approach by employing MediaPipe, which allows real time hand tracking, to automatically record hand coordinate data. This captured data is then input into an LSTM network. This synergy enables the system to more quickly and efficiently process the recognition of hand shapes and gestures. Furthermore, this technique is advantageous in solving practical problems.

3. RESEARCH METHODOLOGY

The main stages in conducting research on recognizing gestures in the ISLS with MediaPipe and LSTM include general steps for data collection, recording ISLS gesture videos, extracting key hand points using MediaPipe, creating an LSTM-based artificial neural network model, and finally training and evaluating the model using a confusion matrix as can be seen in Figure 1. In addition, laying the foundation for automatic real-time gesture detection and recognition through a trained model prototype.

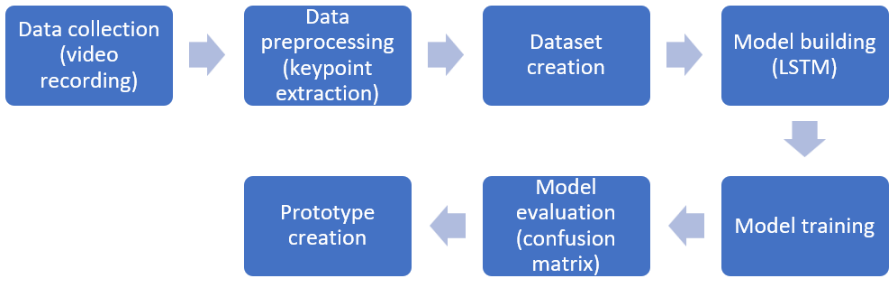

Data collection (video recording)
Data preprocessing (keypoint extraction)
Dataset creation
Model building (LSTM)
Model training
Model evaluation (confusion matrix)
Prototype creation

Figure 1. Research Methodology

3.1. Data Collection

The information utilized in this research is data that was independently gathered by the author. The collected signs consist of 36 signs in the ISLS, which were selected based on the use of one hand, shown in Figure 2(a). This selection of signs aims to simplify the sign recognition process using MediaPipe and LSTM based models.

Each gesture is collected in the form of 25 sequences, with each sequence consisting of 30 images. The selection of the number of sequences per gesture is because most gestures in ISLS do not involve complex movements. Meanwhile, the number of 30 images per sequence is selected based on the average camera frame rate, which is 30 frames per second. The collected data is then arranged in a systematic directory structure, where each folder represents a particular gesture, which is then divided into sequences folders and contains the images in the sequence. This data storage structure can be seen in Figure 2(b).

A, B, C, D, E, F, G, H, I, J, K, L, M,

N, O, P, Q, R, S, T, U, V, W, X, Y, Z,

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

(a)

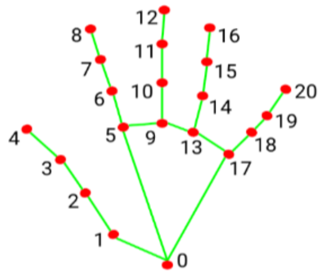
- Signal 1
  - Sequence 1
    - Figure 1
    - Figure 2
    - ...
    - Figure 30
  - ...
  - Sequence 29
  - Sequence 30
- Signal 2
- ...
- Signal n

(b)

Figure 2. (a) Alphabetical and Numerical Signs and (b) Data Storage Structure

3.1.1. Keypoint Extraction

The key point is coordination which indicates the position of the hand in the frame. In this study, keypoint extraction was performed using MediaPipe Hands, a machine learning based hand and finger tracking solution capable of detecting 20 key points or landmarks from just one frame. An illustration of the results of keypoint extraction with MediaPipe is shown in Figure 3.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 3. Landmark Model for Hand

The hand landmark model in MediaPipe generates 21 landmarks representing key points on the hand. Each landmark consists of three main coordinates, namely x, y, and z. Based on the image's width and height, respectively, the x and y coordinates have been normalized into the range [0.0, 1.0]. Meanwhile, the z coordinate represents the depth of the landmark, with the reference point being at the wrist. The closer the landmark is to the camera, the lower the z value. The Z scale has a size that is approximately proportional to the X coordinate scale.

3.1.2. Keypoint Normalization

Since the x, y, and z coordinates are related to the image size, normalization is done for every sequence, while the desired position is the position of the landmark relative to other landmarks. Normalization can be done with the Min-Max scaler. Normalization is done by taking the highest and lowest values of each x, y, and z coordinate. From each coordinate, normalization is performed with the following calculations.

3.1.3. Video Capture

Keypoint extraction requires images as input, where images are the building blocks of a video. Therefore, a video can be considered as a collection of images that can be processed to obtain keypoints, as shown in Figure 4. The video capture process in this study was carried out using OpenCV, an image processing library that allows efficient video recording and manipulation. Each frame in the video recorded by OpenCV will be processed using MediaPipe Hands to generate a hand landmark model. After the hand landmark model is obtained, the keypoints from each frame can be extracted and stored in the form of structured data. This keypoint storage aims to avoid repeating the data collection process, so that the data can be used directly in the model training stage without the need to re-extract from the raw video.

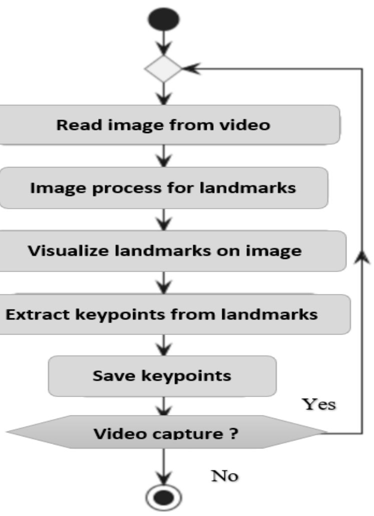

Read image from video
Image process for landmarks
Visualize landmarks on image
Extract keypoints from landmarks
Save keypoints
Video capture ?
Yes
No

Figure 4. Flowchart for Video Capture and Keypoint Extraction

3.2. Long Short-Term Memory Modeling

LSTM is designed to overcome the problem of backpropagation error that often occurs in ordinary RNN. This algorithm was first developed by Hochreiter and Schmidhuber as a modified form of RNN to overcome the difficulty in handling long-term dependencies. Memory cells, input gates, and output gates are the only components of the original LSTM implementation. However, the effectiveness of RNN decreases as the distance in the data sequence increases. To determine which data should be retained or removed from the memory cell, the forget gate layer is added. In this process, xt and $Ct-1$ are used as inputs to determine values between 0 and 1, where '1' means the information is kept, while '0' means the information is deleted. The system then must decide what information will be kept in the memory cell. This stage consists of two main processes: (i) The sigmoid layer is used to select the values that need to be updated. (ii) The tanh layer produces a new vector representing the updated character. The calculation of ft, it, and Ct_hat can be done using the Equations (1), (2), and (3) [44], [45].

$$f_t = \sigma\left(W_f . \left[hidden_{t-1}, dm_{x_t}\right] + b_f\right) \tag{1}$$

$$i_t = \sigma(W_t . [hidden_{t-1}, dm\_x_t] + b_t) \tag{2}$$

$$Ct\_hat = \tanh(W_c . [hidden_{t-1}, dm\_x_t] + b_c) \tag{3}$$

3.2.1. Data Preprocessing

The collected data is loaded and labelled with one-hot category coding values in the form of 0 and 1 in an array. For example, in a data there are three labels where index 0 indicates label A, index 1 indicates label B, and index 2 indicates label C. So, if a datum is label C, its one-hot category coding is, and if a datum is label A, its one-hot category coding is.

3.2.2. Model Creation

The model in this study is used to construct neural network layers. To improve model performance, hyperparameter optimization is performed using the Hyperband algorithm. This model consists of an input layer (30×63), a Dense output layer with softmax activation, and a hidden layer configured using Hyperband. The number of LSTM layers

(1–2), units per layer (128–1024), and dropout rate (0.0–0.5) are among the optimized hyperparameters. Optimization of the batch size (4–256), test size (0.1–0.2), validation size (0.1–0.2), and learning rate (0.0001–0.01) was also carried out. The model was trained using the Adam optimizer, which provides better performance in gradient descent. Because categorical cross-entropy is appropriate for multi-class classification and reduces model errors, it is the loss function that is utilized. The evaluation metric used is categorical accuracy, because it is in accordance with the one-hot encoding format. The model is trained using the collected data, with 100 iterations.

3.2.3. Model Evaluation

The model will predict the test data, then calculate the confusion matrix based on the prediction results and the actual labels. Additionally, each class's F1-score, recall, accuracy, and precision are measured [46].

3.3. Prototype Creation

A prototype can be created using the trained model. In order to be able to read signals continuously, from each frame, a normalized keypoint is obtained and stored in an array, so that if the amount of data in the array is sufficient for the number of images per sequence that has been set, namely 30 images per sequence, then it can be predicted what signals are demonstrated and displayed on the screen.

4. RESULTS AND DISCUSSIONS

The implementation of ISLS hand gesture detection using LSTM is done in Python, supported by various libraries including OpenCV, NumPy, Matplotlib, MediaPipe, Scikit-learn, and TensorFlow. This process is run in Jupyter Notebook, a web-based platform that allows development, documentation, code execution, and interactive communication of results.

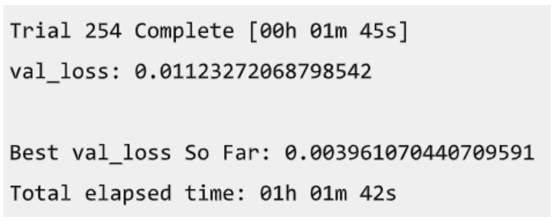
4.1. Model Hyperparameters

Model hyperparameters are parameters configured before training to optimize model performance. The number of LSTM layers, the number of units in each layer, the batch size, the learning rate, the dropout rate, and the percentage of data utilized for training, validation, and testing are among the hyperparameters employed in this model. The selection of the best hyperparameters is done using the Hyperband method, which produces an optimal configuration to improve accuracy and prevent overfitting.

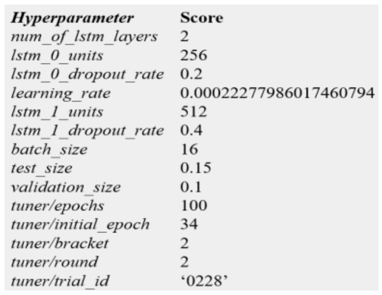
Based on Figure 5(a). The best hyperparameter configuration search results were obtained in the 254th experiment, with the best val_loss value of 0.003961070440709591. This search process lasted for 1 hour 1 minute 42 seconds until the optimal configuration was found. The best hyperparameter configuration is loaded using the command best_hp = tuner.get_best_hyperparameters()[0]. The best hyperparameters obtained from the search process will be used to build and train a model with optimal performance as can be seen in Figure 5(b). The results of the hyperparameter configuration found show a good balance between model capacity and overfitting prevention. The use of two LSTM layers with different dropouts allows the model to better capture temporal patterns, while the small learning rate ensures stable convergence. With evaluations showing high accuracy, this model has great potential to be applied in automatic language gesture recognition.

4.2. Model Building and Training

The model is built using the best hyperparameters with the command model = MyHyperModel().build(best_hp). With this configuration, the resulting model is expected to have optimal performance in detecting hand gestures. Based on Figure 6, the total results of trainable parameters are 1,921,060, indicating that this model is quite complex and able to capture patterns in the data. There are no parameters that cannot be trained, indicating that all layers are active in learning. The combination of two LSTM layers allows the model to understand the long-term relationship between frames, which is very important in gesture recognition. With this configuration, the model can capture hand movement patterns well, but it still needs to be tested to ensure that there is no overfitting to the training data.



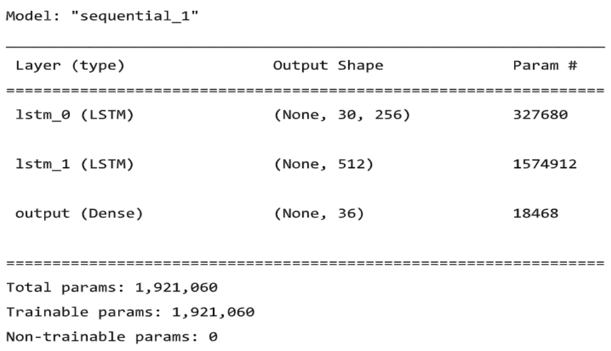
```
Trial 254 Complete [00h 01m 45s]
val_loss: 0.01123272068798542

Best val_loss So Far: 0.003961070440709591
Total elapsed time: 01h 01m 42s
```



Hyperparameter	**Score**
num_of_lstm_layers	2
lstm_0_units	256
lstm_0_dropout_rate	0.2
learning_rate	0.00022277986017460794
lstm_1_units	512
lstm_1_dropout_rate	0.4
batch_size	16
test_size	0.15
validation_size	0.1
tuner/epochs	100
tuner/initial_epoch	34
tuner/bracket	2
tuner/round	2
tuner/trial_id	'0228'

(a) (b)

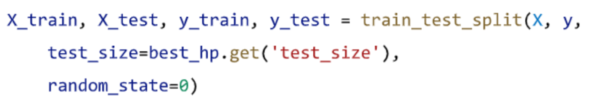
Figure 5. (a) Results of Searching for The Best Hyperparameter Configuration, and (b) Best Hyperparameters



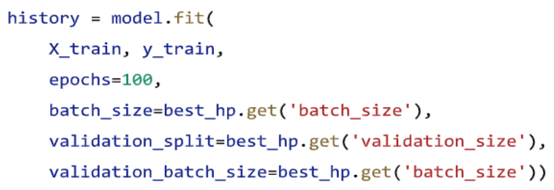
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_0 (LSTM)               (None, 30, 256)           327680

 lstm_1 (LSTM)               (None, 512)               1574912

 output (Dense)              (None, 36)                18468

=================================================================
Total params: 1,921,060
Trainable params: 1,921,060
Non-trainable params: 0
```

Figure 6. Results of The Model with The Best Hyperparameters

The data must then be separated into two categories: training data and testing data, with the proportion of testing data determined based on the value of the test_size hyperparameter. This division ensures that the model can be tested with data that has never been seen before with the commands shown in Figure 7(a) and Figure 7(b).



```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=best_hp.get('test_size'),
    random_state=0)
```

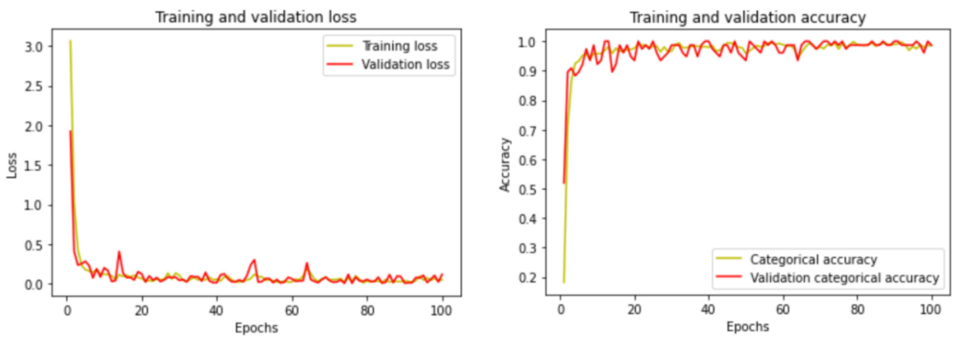


```
history = model.fit(
    X_train, y_train,
    epochs=100,
    batch_size=best_hp.get('batch_size'),
    validation_split=best_hp.get('validation_size'),
    validation_batch_size=best_hp.get('batch_size'))
```

(a) (b)

Figure 7. (a) Separating Training and Testing Data and (b) Train the Model with Training Data

The model is then trained using the same parameters as the hyperparameter search for 100 epochs. To avoid overfitting, cross-validation is used, where the data is divided into training and validation [44], as seen in Figure 8(a) and Figure 8(b). The model is trained using the training data, and its performance on fresh data is assessed using the validation data. Based on Figure 8(a) and Figure 8(b), the model shows a fast-learning process and begins to adapt to the training and validation data since the 10th epoch. After that, the loss decreases slowly, while the accuracy remains stable around a certain value, indicating that the model has reached an optimal point in its learning.



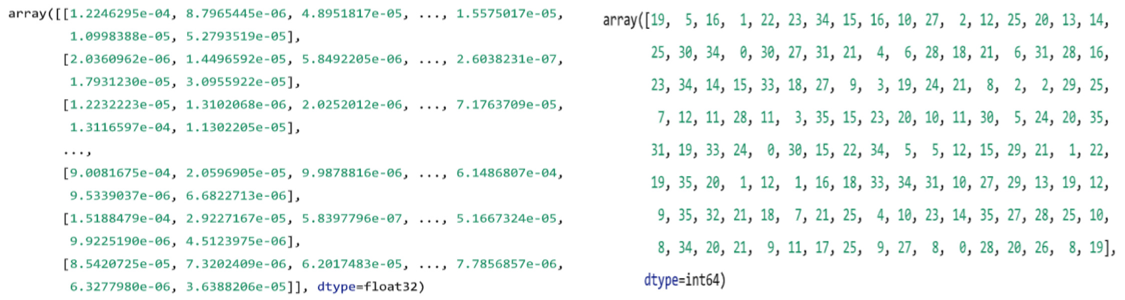
(a) (b)

Figure 8. (a) Training and Validation Loss, and (b) Training and Validation Category Accuracy for The Model with The Best Hyperparameters

4.3. Model Evaluation

Model evaluation is conducted to measure model performance with various evaluation metrics. This process aims to understand the strengths and weaknesses of the model and ensure that the model can work well. This evaluation is also important in monitoring model performance after training. To conduct the evaluation, the model is used to predict previously separated testing data. Prediction is done with the command: pred = model.predict (X_test, batch_size = best_hp.get ('batch_size')). The model's accuracy is then evaluated by comparing the forecast results with the actual labels as shown in Figure 9(a).

To obtain the confusion matrix, the model first predicts the testing data. The prediction results are in the form of a two-dimensional array with the number of rows as many as the testing data and the number of columns according to the number of signal labels. Each element in the array contains the probability value of each class. To be compared with the actual label, the prediction results are converted into labels by taking the index with the highest value using the following command: y_pred = np.argmax(pred, axis=1) as in Figure 9(b).



```
array([[1.2246295e-04, 8.7965445e-06, 4.8951817e-05, ..., 1.5575017e-05,
        1.0998388e-05, 5.2793519e-05],
       [2.0360962e-06, 1.4496592e-05, 5.8492205e-06, ..., 2.6038231e-07,
        1.7931230e-05, 3.0955922e-05],
       [1.2232223e-05, 1.3102068e-06, 2.0252012e-06, ..., 7.1763709e-05,
        1.3116597e-04, 1.1302205e-05],
       ...,
       [9.0081675e-04, 2.0596905e-05, 9.9878816e-06, ..., 6.1486807e-04,
        9.5339037e-06, 6.6822713e-06],
       [1.5188479e-04, 2.9227167e-05, 5.8397796e-07, ..., 5.1667324e-05,
        9.9225190e-06, 4.5123975e-06],
       [8.5420725e-05, 7.3202409e-06, 6.2017483e-05, ..., 7.7856857e-06,
        6.3277980e-06, 3.6388206e-05]], dtype=float32)
```

(a)

```
array([19,  5, 16,  1, 22, 23, 34, 15, 16, 10, 27,  2, 12, 25, 20, 13, 14,
       25, 30, 34,  0, 30, 27, 31, 21,  4,  6, 28, 18, 21,  6, 31, 28, 16,
       23, 34, 14, 15, 33, 18, 27,  9,  3, 19, 24, 21,  8,  2,  2, 29, 25,
        7, 12, 11, 28, 11,  3, 35, 15, 23, 20, 10, 11, 30,  5, 24, 20, 35,
       31, 19, 33, 24,  0, 30, 15, 22, 34,  5,  5, 12, 15, 29, 21,  1, 22,
       19, 35, 20,  1, 12,  1, 16, 18, 33, 34, 31, 10, 27, 29, 13, 19, 12,
        9, 35, 32, 21, 18,  7, 21, 25,  4, 10, 23, 14, 35, 27, 28, 25, 10,
        8, 34, 20, 21,  9, 11, 17, 25,  9, 27,  8,  0, 28, 20, 26,  8, 19],
      dtype=int64)
```

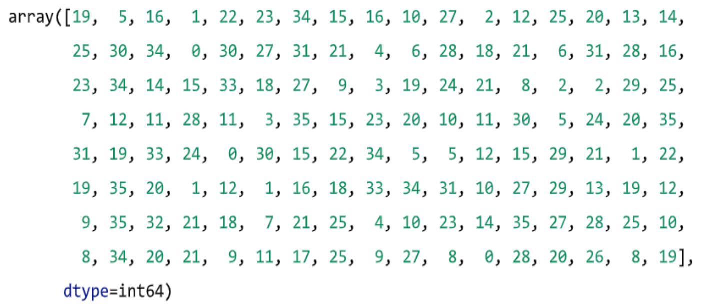
(b)

Figure 9. (a) Testing Data Prediction Results and (b) Results of Taking the Position of the Highest Predicted Value Index for Each Datum from the Predicted Data.

Each data in the prediction results is taken from the index with the highest predicted value to determine the class selected by the model as shown in Figure 10(a). Likewise with the testing data, where the index with the highest value is taken as the actual label. This process is carried out with the following command: y_true = np.argmax (y_test, axis =1). With these results, the model can be compared with the actual data to evaluate its performance using the confusion matrix and other evaluation metrics.

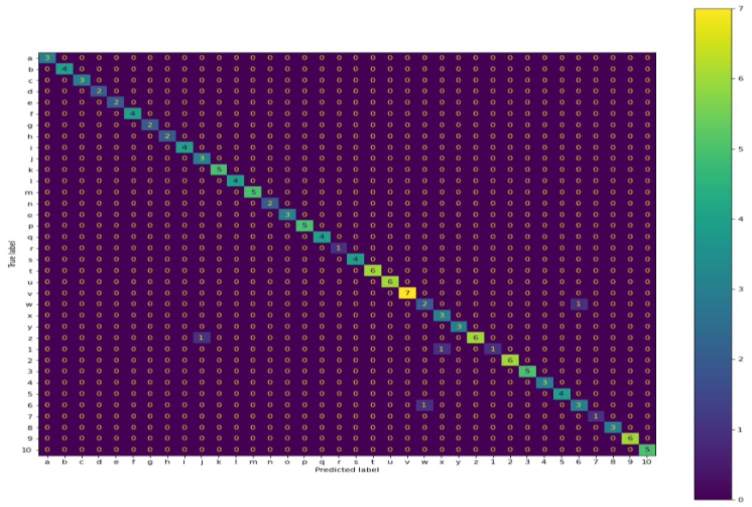
4.3.1 Confusion Matrix

A confusion matrix is developed to assess how well the model performs when it comes to hand gesture classification. From a total of 136 testing data, there is 1 misclassification in each of the four categories, while the rest are correctly classified. This demonstrates how accurately the model can identify hand motions, although there are still a few prediction errors, shown in Figure 10(b).



```
array([19,  5, 16,  1, 22, 23, 34, 15, 16, 10, 27,  2, 12, 25, 20, 13, 14,
       25, 30, 34,  0, 30, 27, 31, 21,  4,  6, 28, 18, 21,  6, 31, 28, 16,
       23, 34, 14, 15, 33, 18, 27,  9,  3, 19, 24, 21,  8,  2,  2, 29, 25,
        7, 12, 11, 28, 11,  3, 35, 15, 23, 20, 10, 11, 30,  5, 24, 20, 35,
       31, 19, 33, 24,  0, 30, 15, 22, 34,  5,  5, 12, 15, 29, 21,  1, 22,
       19, 35, 20,  1, 12,  1, 16, 18, 33, 34, 31, 10, 27, 29, 13, 19, 12,
        9, 35, 32, 21, 18,  7, 21, 25,  4, 10, 23, 14, 35, 27, 28, 25, 10,
        8, 34, 20, 21,  9, 11, 17, 25,  9, 27,  8,  0, 28, 20, 26,  8, 19],
      dtype=int64)
```

(a)



(b)

Figure 10. (a) Results of Taking the Position of the Highest Predicted Value Index for Each Datum from the Testing Data, and (b) Confusion Matrix

4.3.2 Accuracy, Precision, Recall, F1-score, and Support

The sklearn classification report function is utilized to display model evaluation metrics, namely accuracy, precision, recall, F1-score, and support with the classification report (y_true, y_pred, target_names=GESTURES, digits=3) command with the results in Table 1. Accuracy measures how well the model classifies all classes. Precision displays the proportion of accurate forecasts to all predictions for a class, while recall measures how much data is correctly identified by the model. The F1-score strikes a compromise between recall and precision. Support shows the number of correct samples in each target class.

Based on Figure 11, the model has high accuracy (97.1%), indicating that most hand gestures can be recognized well. Some classes still have misclassifications, especially the letters “W”, “X”, “Z”, and the numbers “1” and “6”. Recall is lower than precision, which means the model is more likely to avoid misclassification but still has some classes that are difficult to recognize. Improvements are needed in the data and model, for example by increasing the amount of data in classes that have low recall or using data augmentation techniques to improve model generalization.

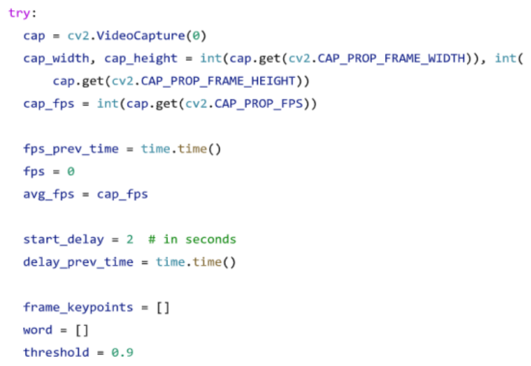
	Precision	*Recall*	*F1-score*	*Support*		*Precision*	*Recall*	*F1-score*	*Support*
A	100%	100%	100%	3	U	100%	100%	100%	6
B	100%	100%	100%	4	V	100%	100%	100%	7
C	100%	100%	100%	3	W	66.7%	66.7%	66.7%	3
D	100%	100%	100%	2	X	75.0%	100%	85.7%	3
E	100%	100%	100%	2	Y	100%	100%	100%	3
F	100%	100%	100%	4	Z	100%	85.7%	92.3%	7
G	100%	100%	100%	2	1	100%	50.0%	66.7%	2
H	100%	100%	100%	2	2	100%	100%	100%	6
I	100%	100%	100%	4	3	100%	100%	100%	5
J	75.0%	100%	85.7%	3	4	100%	100%	100%	3
K	100%	100%	100%	5	5	100%	100%	100%	4
L	100%	100%	100%	4	6	75.0%	75.0%	75.0%	4
M	100%	100%	100%	5	7	100%	100%	100%	1
N	100%	100%	100%	2	8	100%	100%	100%	3
O	100%	100%	100%	3	9	100%	100%	100%	6
P	100%	100%	100%	5	10	100%	100%	100%	5
Q	100%	100%	100%	4	Accuracy	97.1%			136
R	100%	100%	100%	1	Macro avg.	97.0%	96.6%	96.4%	136
S	100%	100%	100%	4	Weighted avg.	97.4%	97.1%	97.0%	136
T	100%	100%	100%	6					

Figure 11. Results of The Classification Report from The Prediction and Testing Data

4.4. Prototype Creation

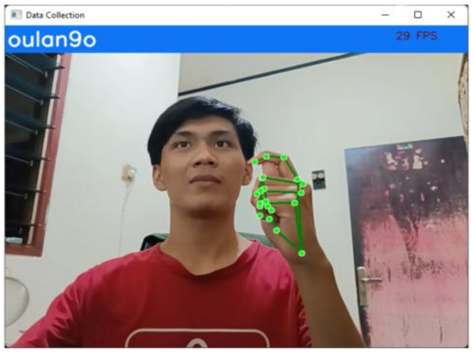
A prototype is created to test the trained model and as a basis for further research using the command in Figure 12(a) truncated script. In this prototype, each gesture must consist of 30 frames. If the webcam has a frame rate of 30 fps, then one gesture can be done in one second. However, because the shooting speed can vary, for example only 14 fps, filler is needed to complete the number of frames up to 30 per sequence.

A prototype was created to arrange letters and numbers into sequences that form words, Figure 12(b). This prototype gives a beep sound as a sign that the sequence capture is about to begin. A two-second pause is given so that the user can prepare. After that, the sequence is recorded for one second, then the model predicts the gestures made based on the data that has been collected.



```
try:
  cap = cv2.VideoCapture(0)
  cap_width, cap_height = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(
      cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
  cap_fps = int(cap.get(cv2.CAP_PROP_FPS))

  fps_prev_time = time.time()
  fps = 0
  avg_fps = cap_fps

  start_delay = 2  # in seconds
  delay_prev_time = time.time()

  frame_keypoints = []
  word = []
  threshold = 0.9
```

(a)



(b)

Figure 12. (a) Code for Prototype, and (b) Prototypes to Form a Word

5. CONCLUSION

This study successfully produced an LSTM-based neural network model that can classify alphabets and numbers in the ISLS using MediaPipe. The developed model has an optimal hyperparameter configuration found through the Hyperband method, with two LSTM layers and an adjusted dropout rate. Model evaluation showed high accuracy, reaching 97.1%, with precision, recall, and F1-score also approaching that number. In addition, a prototype was successfully created as a basis for implementing this model in a wider system. For further development, it is recommended to use a more accurate hand recognition framework than MediaPipe, improve data pre-processing so that the differences between gestures are clearer, and explore other DL methods such as SSD, YOLO, or Faster-RCNN. Pose and facial recognition can also be included to increase the model's accuracy in more thoroughly identifying ISLS gestures.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

FUNDING STATEMENT

The authors received no funding from any party for the research and publication of this article.

AUTHOR CONTRIBUTIONS

Wargijono Utomo: Draft Preparation, Conceptualization, Writing - Editing;
Yogasetya Suhanda: Data Acquisition – Preprocessing, Writing;
Harun Ar-Rasyid: Modeling – Writing.
Andy Dharmalau: Methodology, Supervision, Review – Editing.

CONFLICT OF INTERESTS

No conflict of interests were disclosed.

ETHICS STATEMENTS

Our publication ethics follow The Committee of Publication Ethics (COPE) guideline. https://publicationethics.org/

REFERENCES

[1] A.R. Syulistyo, D.S. Hormansyah, and P.Y. Saputra, “SIBI (Sistem Isyarat Bahasa Indonesia) translation using Convolutional Neural Network (CNN),” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 732, no. 1, 2020, doi: 10.1088/1757-899X/732/1/012082.

[2] A. Aljabar and Suharjito, “BISINDO (Bahasa isyarat indonesia) sign language recognition using CNN and LSTM,” *Adv. Sci. Technol. Eng. Syst.*, vol. 5, no. 5, pp. 282–287, 2020, doi: 10.25046/AJ050535.

[3] M. Al-Qurishi, T. Khalid, and R. Souissi, “Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues,” *IEEE Access*, vol. 9, pp. 126917–126951, 2021, doi: 10.1109/ACCESS.2021.3110912.

[4] A. M. Buttar, U. Ahmad, A. H. Gumaei, A. Assiri, M.A. Akbar, and B.F. Alkhamees, “Deep Learning in Sign Language Recognition: A Hybrid Approach for the Recognition of Static and Dynamic Signs,” *Mathematics*, vol. 11, no. 17, pp. 1–20, 2023, doi: 10.3390/math11173729.

[5] Z. Zhou, V.W.L. Tam, and E.Y. Lam, “SignBERT: A BERT-Based Deep Learning Framework for Continuous Sign Language Recognition,” *IEEE Access*, vol. 9, pp. 161669–161682, 2021, doi: 10.1109/ACCESS.2021.3132668.

[6] Z. Zhang, Y. Tang, S. Zhao, and X. Zhang, “Real-time surface EMG pattern recognition for hand gestures based on support vector machine,” *IEEE Int. Conf. Robot. Biomimetics, ROBIO 2019*, pp. 1258–1262, 2019, doi: 10.1109/ROBIO49542.2019.8961436.

[7] M. Haroon, S. Altaf, S. Ahmad, M. Zaindin, S. Huda, and S. Iqbal, “Hand Gesture Recognition with Symmetric Pattern under Diverse Illuminated Conditions Using Artificial Neural Network,” *Symmetry (Basel).*, vol. 14, no. 10, 2022, doi: 10.3390/sym14102045.

[8] Y. Wang, Y. Zhao, and S. Addepalli, “Practical Options for Adopting Recurrent Neural Network and Its Variants on Remaining Useful Life Prediction,” *Chinese J. Mech. Eng. (English Ed.*, vol. 34, no. 1, 2021, doi: 10.1186/s10033-021-00588-x.

[9] M. Zulqarnain, R. Ghazali, M. G. Ghouse, Y.M.M. Hassim, and I. Javid, “Predicting Financial Prices of Stock Market using Recurrent Convolutional Neural Networks,” *Int. J. Intell. Syst. Appl.*, vol. 12, no. 6, pp. 21–32, 2020, doi:

10.5815/ijisa.2020.06.02.

[10] J. Liu and X. Gong, "Attention mechanism enhanced LSTM with residual architecture and its application for protein-protein interaction residue pairs prediction," *BMC Bioinformatics*, vol. 20, no. 1, pp. 1–11, 2019, doi: 10.1186/s12859-019-3199-1.

[11] Q. Ma, Z. Lin, E. Chen, and G.W. Cottrell, "Temporal pyramid recurrent neural network," *AAAI 2020 - 34th AAAI Conf. Artif. Intell.*, pp. 5061–5068, 2020, doi: 10.1609/aaai.v34i04.5947.

[12] C. Avci, B. Tekinerdogan, and C. Catal, "Analyzing the performance of long short-term memory architectures for malware detection models," *Concurr. Comput. Pract. Exp.*, vol. 35, no. 6, p. 1, 2023, doi: 10.1002/cpe.7581.

[13] H.M. Radha, A.K.A. Hassan, and A.H. Al-Timemy, "Enhanced Prosthesis Control Through Improved Shoulder Girdle Motion Recognition Using Time-Dependent Power Spectrum Descriptors and Long Short-Term Memory," *Math. Model. Eng. Probl.*, vol. 10, no. 3, pp. 861–870, 2023, doi: 10.18280/mmep.100316.

[14] M.H. Ismail, S.A. Dawwd, and F.H. Ali, "Dynamic hand gesture recognition of Arabic sign language by using deep convolutional neural networks," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 25, no. 2, pp. 952–962, 2022, doi: 10.11591/ijeecs.v25.i2.pp952-962.

[15] J.R. Gonzalez-Rodriguez, D.M. Cordova-Esparza, J.Terven, and J.A. Romero-Gonzalez, "Towards a Bidirectional Mexican Sign Language–Spanish Translation System: A Deep Learning Approach," *Technologies*, vol. 12, no. 1, pp. 1–16, 2024, doi: 10.3390/technologies12010007.

[16] M. De Coster, E. Rushe, R. Holmes, A. Ventresque, and J. Dambre, "Towards the extraction of robust sign embeddings for low resource sign language recognition," 2023. [Online]. Available: http://arxiv.org/abs/2306.17558.

[17] G.H. Samaan *et al.*, "MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition," *Electron.*, vol. 11, no. 19, pp. 1–15, 2022, doi: 10.3390/electronics11193228.

[18] J. Shin, A.S.M. Miah, K. Suzuki, K. Hirooka, and M.A.M. Hasan, "Dynamic Korean Sign Language Recognition Using Pose Estimation Based and Attention-Based Neural Network," *IEEE Access*, vol. 11, no. December, pp. 143501–143513, 2023, doi: 10.1109/ACCESS.2023.3343404.

[19] D.K. Singh, "3D-CNN based Dynamic Gesture Recognition for Indian Sign Language Modeling," *Procedia CIRP*, vol. 189, pp. 76–83, 2021, doi: 10.1016/j.procs.2021.05.071.

[20] M. Kakizaki, A.S.M. Miah, K. Hirooka, and J. Shin, "Dynamic Japanese Sign Language Recognition Throw Hand Pose Estimation Using Effective Feature Extraction and Classification Approach," *Sensors*, vol. 24, no. 3, 2024, doi: 10.3390/s24030826.

[21] X. Dang, W. Li, J. Zou, B. Cong, and Y. Guan, "Assessing the impact of body location on the accuracy of detecting daily activities with accelerometer data," *iScience*, vol. 27, no. 2, pp. 108626, 2024, doi: 10.1016/j.isci.2023.108626.

[22] M.A. Khatun *et al.*, "Deep CNN-LSTM With Self-Attention Model for Human Activity Recognition Using Wearable Sensor," *IEEE J. Transl. Eng. Heal. Med.*, vol. 10, no. May, pp. 1–16, 2022, doi: 10.1109/JTEHM.2022.3177710.

[23] H. Heydarian, P.V. Rouast, M.T.P. Adam, T. Burrows, C.E. Collins, and M.E. Rollo, "Deep learning for intake gesture detection from wrist-worn inertial sensors: The effects of data preprocessing, sensor modalities, and sensor positions," *IEEE Access*, vol. 8, pp. 164936–164949, 2020, doi: 10.1109/ACCESS.2020.3022042.

[24] J. Homepage, I. Gusti, A. Oka Aryananda, and F. Samopa, "Comparison of the Accuracy of The Bahasa Isyarat Indonesia (BISINDO) Detection System Using CNN and RNN Algorithm for Implementation on Android," *MALCOM: Indonesian Journal of Machine Learning and Computer Science 4*, no. 3, pp. 1111-1119, 2024, doi: 10.57152/malcom.v4i3.1465.

[25] I.D.M.B. Atmaja Darmawan, Linawati, G. Sukadarmika, N.M.A.E.D. Wirastuti, and R. Pulungan, "Temporal Action Segmentation in Sign Language System for Bahasa Indonesia (SIBI) Videos Using Optical Flow-Based Approach," *J. Ilmu Komput. dan Inf.*, vol. 17, no. 2, pp. 195–202, 2024, doi: 10.21609/jiki.v17i2.1284.

[26] F. Wijaya, L. Dahendra, E.S. Purwanto, and M.K. Ario, "Quantitative analysis of sign language translation using artificial neural network model," *Procedia Comput. Sci.*, vol. 245, no. C, pp. 998–1009, 2024, doi: 10.1016/j.procs.2024.10.328.

[27] A.S.M. Miah, M.A.M. Hasan, S. Nishimura, and J. Shin, "Sign Language Recognition Using Graph and General Deep Neural Network Based on Large Scale Dataset," *IEEE Access*, vol. 12, no. February, pp. 34553–34569, 2024, doi: 10.1109/ACCESS.2024.3372425.

[28] D. Kumari, and R.S. Anand, "AC2_Fiestas_Axel," *Electron. 2024, Vol. 13, Page 1229*, vol. 13, no. 7, pp. 1229, 2024.

[29] A.S.M. Miah, M.A.M. Hasan, Y. Tomioka, and J. Shin, "Hand Gesture Recognition for Multi-Culture Sign Language Using Graph and General Deep Learning Network," *IEEE Open J. Comput. Soc.*, vol. 5, no. February, pp. 144–155, 2024, doi: 10.1109/OJCS.2024.3370971.

[30] S. Arooj, S. Altaf, S. Ahmad, H. Mahmoud, and A.S.N. Mohamed, "Enhancing sign language recognition using CNN and SIFT: A case study on Pakistan sign language," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 36, no. 2, pp. 101934, 2024, doi: 10.1016/j.jksuci.2024.101934.

[31] N. Sign, and L. Recognition, "Norwegian Sign Language Recognition," 2023.

[32] J.P. Sahoo, A.J. Prakash, P. Plawiak, and S. Samantray, "Real-time hand gesture recognition using fine-tuned convolutional neural network, " *Sensors*, 22(3), 706, 2022, doi: 10.3390/s22030706.

[33] M.A. Rahim, J. Shin, and K.S. Yun, "Hand gesture-based sign alphabet recognition and sentence interpretation using a convolutional neural network," *Ann. Emerg. Technol. Comput.*, vol. 4, no. 4, pp. 20–27, 2020, doi: 10.33166/AETiC.2020.04.003.

[34] W. Jintanachaiwat *et al.*, "Using LSTM to translate Thai sign language to text in real time," *Discov. Artif. Intell.*, vol. 4, no. 1, 2024, doi: 10.1007/s44163-024-00113-8.

[35] Q.M. Areeb, M. Nadeem, R. Alroobaea, and F. Anwer, "Helping Hearing-Impaired in Emergency Situations : A Deep Learning-Based Approach," *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3142918.

[36] D. Das Chakladar, P. Kumar, S. Mandal, P.P. Roy, M. Iwamura, and B.G. Kim, "3D Avatar Approach for Continuous Sign Movement Using Speech/Text," *Appl. Sci.*, vol. 11, no. 8, 2021, doi: 10.3390/app11083439.

[37] I. Jayaweerage *et al.*, "Motion Capturing in cricket with bare minimum hardware and optimised software: A comparison of MediaPipe and OpenPose," *2024 1st Int. Conf. Software, Syst. Inf. Technol. SSITCON 2024*, December, 2024, doi: 10.1109/SSITCON62437.2024.10796169.

[38] M. Latyshev, G. Lopatenko, V. Shandryhos, O. Yarmoliuk, M. Pryimak, and I. Kvasnytsia, "Computer Vision Technologies for Human Pose Estimation in Exercise: Accuracy and Practicality," *Soc. Integr. Educ. Proc. Int. Sci. Conf.*, vol. 2, pp. 626–636, 2024, doi: 10.17770/sie2024vol2.7842.

[39] N.H.M. Dhuzuki *et al.*, "Design and Implementation of a Deep Learning Based Hand Gesture Recognition System for Rehabilitation Internet-of-Things (Riot) Environments Using Mediapipe," *IIUM Eng. J.*, vol. 26, no. 1, pp. 353–372, 2025, doi: 10.31436/IIUMEJ.V26I1.3455.

[40] J.M. Joshi, and D.U. Patel, "Dynamic Indian Sign Language Recognition Based on Enhanced LSTM with Custom Attention Mechanism," *SSRG Int. J. Electron. Commun. Eng.*, vol. 11, no. 2, pp. 60–68, 2024, doi: 10.14445/23488549/IJECE-V11I2P107.

[41] A. Sugiharto *et al.*, "Comparison of SVM, Random Forest and KNN Classification By Using HOG on Traffic Sign Detection," *2022 6th Int. Conf. Informatics Comput. Sci.*, pp. 60–65, 2022, doi: 10.1109/ICICoS56336.2022.9930588.

[42] A.G. Mahmoud, A.M. Hasan, and N.M. Hassan, "Convolutional neural networks framework for human hand gesture recognition," *Bull. Electr. Eng. Informatics*, vol. 10, no. 4, pp. 2223–2230, 2021, doi: 10.11591/EEI.V10I4.2926.

[43] I.D. Mienye, T.G. Swart, and G. Obaido, "Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications," *Information*, vol. 15, no. 9, pp. 517, 2024, doi: 10.3390/info15090517.

[44] M.R. Abdurrahman, H. Al-aziz, F.A. Zayn, and M. Agus, "Development of Robot Feature for Stunting Analysis Using Long-Short Term Memory (LSTM) Algorithm," *J. Informatics Web Eng.*, 3(3), pp. 164–175, doi: 10.33093/jiwe.2024.3.3.10.

[45] S.K. Paul *et al.*, "An Adam based CNN and LSTM approach for sign language recognition in real time for deaf people," *Bull. Electr. Eng. Informatics*, vol. 13, no. 1, pp. 499–509, 2024, doi: 10.11591/eei.v13i1.6059.

[46] A. Khan, K. Khan, W. Khan, S.N. Khan, and R. Haq, "Knowledge-based Word Tokenization System for Urdu," *J. Informatics Web Eng.*, vol. 3, no. 2, pp. 86–97, 2024, doi: 10.33093/jiwe.2024.3.2.6.

BIOGRAPHIES OF AUTHORS

	Wargijono Utomo is a lecturer at Department of Information System and Informatics, Faculty of Engineering, Krisnadwipayana University. He is also a researcher at the research and community service center. His is research interests include data mining, machine learning and deep learning. He can be contacted at email: wargiono@unkris.ac.id.
	Yogasetya Suhanda is a lecturer at Department of Information System, Faculty of Technology, Swadharma Institute of Technology and Business. His research interests include Information Systems, Information Technology Management, Data mining. He can be contacted at email: yogasetyas@swadharma.ac.id
	Harun Ar-Rasyid is a lecturer at Department of Data Science, Faculty of Technology, Swadharma Institute of Technology and Business. His research interests include Information Systems and Data mining. He can be contacted at email: harun@swadharma.ac.id
	Andy Dharmalau is a lecturer at Department of Informatics Engineering, Faculty of Technology, Swadharma Institute of Technology and Business. His research interests include Information Systems, IoT Technology and Data Mining. He can be contacted at email: andy.d@swadharma.ac.id