

Temporal convolutional network for Indonesian word by word sign language translator on MediaPipe framework

Sarah Purnamawati¹, Albert¹, Anggi Yohanes Pardede¹, Iqbal Fakhri¹, Farhad Nadi²,
Mohammad Fadly Syahputra¹, Romi Fadillah Rahmat¹

¹Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

²Faculty of Artificial Intelligence and Frontier Technologies, UNITAR International University, Petaling Jaya, Malaysia

Article Info

Article history:

Received Jul 12, 2025

Revised Aug 4, 2026

Accepted Aug 12, 2026

Keywords:

Deep learning

Hand movement recognition

Language automation

Neural network

Speech-impaired technology

ABSTRACT

Sign language is a language created to help the deaf-mute communicate with others. In Indonesia, there are two sign languages: Indonesian sign language system or *sistem isyarat bahasa Indonesia* (SIBI) and Indonesian sign language or *bahasa isyarat Indonesia* (BISINDO), but the deaf community prefers to use BISINDO. Deaf-mute people use sign language to communicate, but normal people usually avoid learning it because they feel it's unnecessary for daily communication. This will certainly be a problem for the deaf-mute because people who do not learn sign language will certainly have difficulty understanding what they are trying to convey. Therefore, an application that can translate sign language is needed to help overcome communication problems between the deaf-mute and normal people. This research focuses on BISINDO translation using a temporal convolutional network (TCN) with MediaPipe methods. The amount of data used consists of 2,700 videos representing 27 word-level signs in BISINDO; this number increases to 32,400 after undergoing the data augmentation process. The model built was tested in translating BISINDO at the word level in simulation, and a thorough comparison between TCN and bidirectional long short-term memory (BiLSTM); both of them were running on the MediaPipe platform separately. From the experiment, it is shown that the performance of TCN exceeds BiLSTM with an accuracy rate as high as 88.64%.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Sarah Purnamawati

Department of Information Technology, Faculty of Computer Science and Information Technology

Universitas Sumatera Utara

Medan, Indonesia

Email: sarah_purnamawati@usu.ac.id

1. INTRODUCTION

As social beings, it is human nature to socialize and interact with other humans through communication to obtain the information needed to adapt to the surrounding environment. However, the communication process can be disrupted due to factors such as hearing loss or limited speaking ability suffered by one of the parties in a communication. Due to this difficulty, a language was created to help the communication process, namely sign language. Sign language is a form of human non-verbal communication expressed through hand gestures to visually convey information to recipients [1]. Sign language usually emerges naturally and adapts to the culture of each region; this makes each region have its own sign language. For example, America uses American sign language (ASL) [2], the United Kingdom uses British sign language (BSL) [3], and India uses Indian sign language (ISL) [4]. Indonesia itself has two sign

languages in use, namely Indonesian sign language system or *sistem isyarat bahasa Indonesia* (SIBI) and Indonesian sign language or *bahasa isyarat Indonesia* (BISINDO). SIBI was engineered by hearing individuals as a standardized system [5]. Currently, SIBI is designated by the government as the official sign language in Indonesia and is taught in special schools there. BISINDO is a sign language formed by the deaf community in Indonesia and emerged naturally based on the observations of the deaf [6]. The term "BISINDO" was only used in 2016 in a resolution of the seventh Gerkatin congress held in Makassar [7]. Although SIBI is an official sign language in Indonesia, in fact, deaf-mute people prefer to use BISINDO rather than SIBI. Deaf-mute people feel that BISINDO is easier to use for communication because it is a natural sign language. The interaction between facial emotions and natural motion gestures improves language understanding for both deaf and hearing individuals [8].

Previous research on sign language recognition has been conducted by various researchers [9]–[13]. These studies have used various models, such as extreme learning machine (ELM) [14], convolutional neural network (CNN), long short-term memory (LSTM), and a combination of these, to analyze various hand signal images. Kodandaram *et al.* [9] used three CNN models on the ASL dataset, which resulted in 97% accuracy for the LeNet-5 model, 98.9% accuracy for the MobileNetV2 model, and 98% accuracy for the built model. Singh [10] used the 3D CNN method on a dataset consisting of 20 ISL hand gestures, which were captured under different environmental conditions, such as lighting and background conditions. The videos were then framed and extracted, and 162,840 images were obtained. The average accuracy obtained was as high as 88.24%. Coster *et al.* [11] used OpenPose and a transformer architecture to recognize the Flemish sign language or *Vlaamse gebarentaal* (VGT) dataset, which consisted of 100 classes consisting of 104 sign words. This research trained three transformer models, each with a different embedding size, and achieved an accuracy value of 74.7%. Tunga *et al.* [12] used a graph convolutional network (GCN) and bidirectional encoder representations from transformers (BERT) to translate ASL, with an accuracy value of 60.15% and 42.18% on the first dataset and 42.18% on the second dataset. Ghosh [15] used MediaPipe as a cross-platform framework, combining it with LSTM to translate ASL. The best performing algorithm in the test was SVM, with accuracy values of 99.15%, 99.29%, 98.19%, 99.18%, and 96.22% in the five datasets, respectively. Aljabar and Suhajito [16] used three models, namely CNN, LSTM, and a combination of CNN and LSTM, on the BISINDO dataset. The average accuracy obtained by the CNN model was 73%, the LSTM model was 81%, and the CNN-LSTM model was 90% in testing. Halder and Tayade [17] conducted research on sign language recognition from various countries, including America, India, Italy, and Turkey, using a support vector machine (SVM) combined with MediaPipe for gesture feature extraction. Tsinganos *et al.* [18] proposed the use of temporal convolutional networks (TCNs) to address the problem of surface electromyography (sEMG)-based hand gesture recognition. This study highlights the importance of developing tailored data feeding methods to improve the performance of TCN models in more dynamic usage situations. Ishaq *et al.* [19] developed a TCN-based emotion recognition system that utilizes a spatio-TCN to ascertain the speaker's emotional state from speech signals. The system automatically extracts valid sequence cues from speech signals, outperforming both traditional and state-of-the-art (SOTA) machine learning algorithms.

Based on previous research, it can be seen that the main reasons for using MediaPipe [15], [17] compared to other frameworks such as OpenPose [11] are related to several things. Among them is the superior ease of integration for MediaPipe. In addition, MediaPipe is superior in the mobile environment compared to OpenPose, and MediaPipe also has fewer keypoints compared to OpenPose. While MediaPipe and OpenPose are more on the library side, there also exist sign language applications that utilize the hardware section, such as the leap motion device [20] and edge device [21]. The pre-processing stage in Coster *et al.* [11] did not include steps such as frame selection and normalization. Similarly, Tunga *et al.* [12] used GCN and BERT to translate ASL without any data pre-processing. The authors [15], [17] used MediaPipe as a modelling framework. However, both of these studies were limited to modeling the hand region. Thus, the work models the entire body, hands, and face. Additionally, their classification methods and datasets differ, and they did not perform any data pre-processing. This study differs from Aljabar and Suhajito [16] in its algorithm and pre-processing stages. Their study employed three models: CNN, LSTM, and a hybrid CNN-LSTM with pre-processing that included background subtraction, grayscaling, and resizing. Likewise, the research by Dwijayanti *et al.* [6] employed a different CNN algorithm and performed only image resizing during pre-processing, focusing only on static alphabet signs. The main differences between this study and several related studies that have been conducted are in the method, dataset, data pre-processing method, and the type of translated language. In this study, the algorithms used are bidirectional long short-term memory (BiLSTM) [22], [23], TCN [18], and MediaPipe [24]. Previous studies also indicated that MediaPipe has been widely used in sign language detection, particularly in identifying hand and facial landmarks. However, the majority of studies still confine themselves to static gesture recognition or rely on image-based classification models. The main difference from previous studies is the focus on real-time video-based

gesture recognition, rather than just static images, so that the developed system can capture the complexity of hand gestures in real communication.

2. METHOD

There are several stages carried out in this research in order to realize the objectives of the research. Figure 1 shows the system architecture of this research. The stages carried out in this research are video acquisition, video preprocessing, feature extraction, data preprocessing, the training process, and the testing process. At the video acquisition stage, data in the form of BISINDO gesture videos are collected, then cleaned and resized as needed. Furthermore, the video goes through a preprocessing stage which includes the process of frame extraction, keyframe selection, and reducing the number of frames, before keypoint extraction is carried out at the feature extraction stage. In the feature extraction stage, the extracted keypoints will go through augmentation and normalization stages to increase the generalization level of the model. The extracted keypoints are then processed again to produce input data that will be used when training the model. The training stage will produce a trained model that will be used to translate BISINDO in the testing stage. The testing process is carried out in real time to test the generalization of the model in translating BISINDO in several different situations.

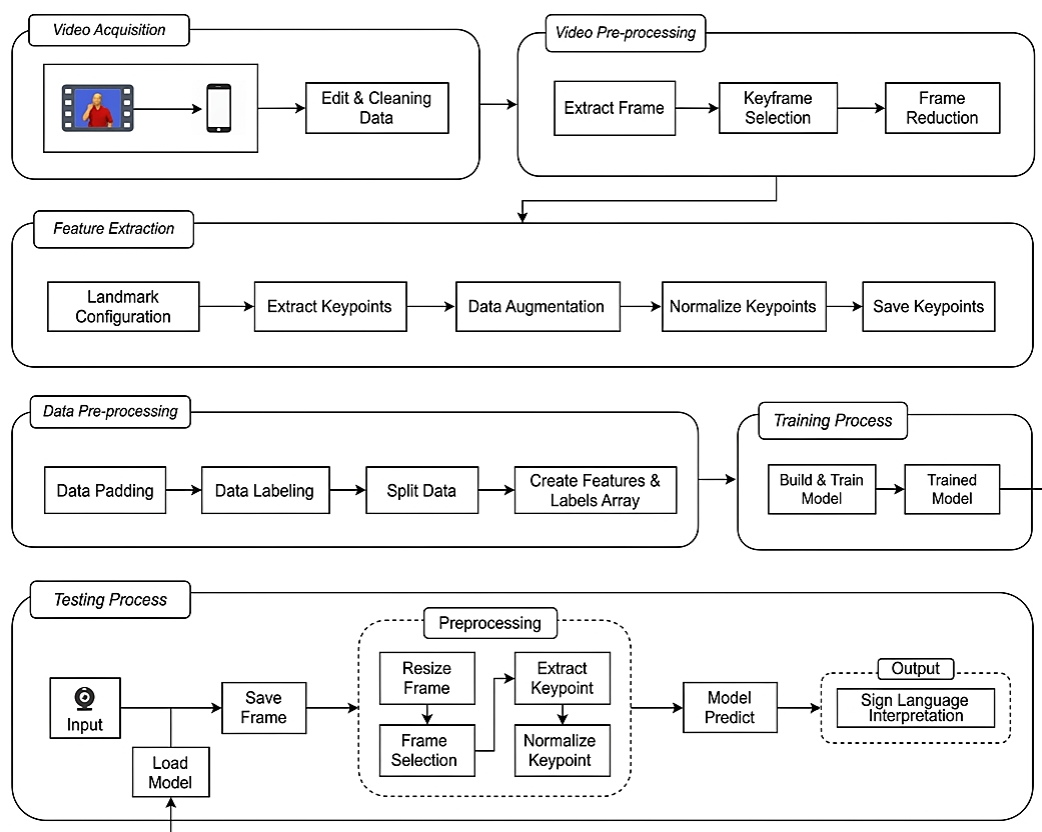


Figure 1. System architecture

2.1. Dataset

The data used are word-level BISINDO signs taken using a smartphone camera with a resolution of 14 MP and a frame rate of 30 frames per second, mounted on a tripod. The tripod was placed at a distance of approximately 75 cm from the signer, with a height of approximately 1.4 meters. The video was taken in mid shot, from the top of the head to the waist (see Figure 2). Ensure the entire body (from waist to top of head) appears in each video frame as shown in Figure 2(a). This figure demonstrates the requirement to keep the entire body (from waist to top of head) within the frame. As shown in Figure 2(b), the gesture movements must be clear and in frame. This figure highlights the incorrect cut-off of the hand during a sign. Recording must be done in well-lit conditions, as shown in Figure 2(c), to ensure clear capture of key points. This figure

contrasts proper illumination with an underexposed, dark environment. Video recordings must have a stable background and not include any hands or faces apart from the hands and face of the performer. Adjust the recording device so that it is as stable and focused as possible. The recorded video was in .mp4 format with a frame size of 1920×1080 pixels and was a long video consisting of several sign videos. The raw video is then cleaned and cut, the number of sign videos obtained from cutting the raw video is 2,700 videos consisting of 27 words, which means each sign has a total of 100 videos. Each video is then put into a directory with a name that corresponds to the meaning of its respective sign. We gathered all the data from 10 people—5 males and 5 females— including those from the deaf-mute school. The participants have been creating several videos demonstrating each of the 27 sign language words along with their various interpretations. Thus, it means every person will perform 10 sign language videos for every 27 words. The data was distributed into two groups: training and validation, while testing will be done in real time. The training data will take 70% of the overall data, and the validation will take 30% of the rest. The 27 signs used can be grouped into six categories:

- i) Question words: 'what', 'where', 'how much', 'when', 'who', 'why', and 'how'.
- ii) Expressions: 'thank you' and 'sorry'.
- iii) Greetings: 'good morning', 'good afternoon', and 'good evening'.
- iv) Time words: 'today', 'tomorrow', 'now', 'yesterday', and 'day'.
- v) Personal pronouns: 'I', 'he/she', 'you', and 'we'.
- vi) Additional words: 'name', 'number', 'phone', 'deaf', 'eat', and 'sir'.

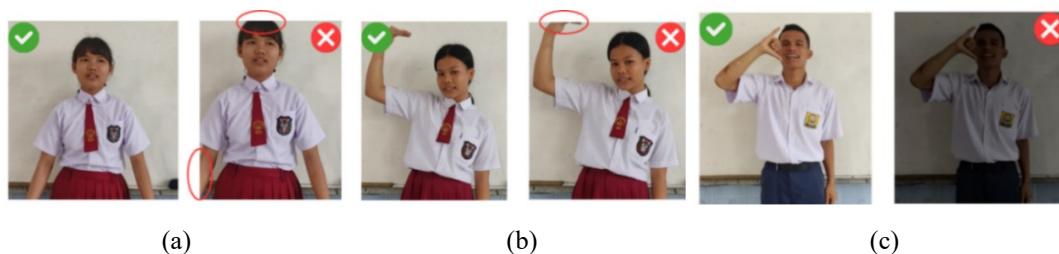


Figure 2. Correct and incorrect image acquisition samples: (a) body framing, (b) gesture movement, and (c) lighting conditions

2.2. Video preprocessing: frame extraction, keyframe selection, and frame reduction

There are three processes in the video preprocessing stage, frame extraction, keyframe selection, and frame reduction. The number of frames extracted from 2,700 videos is 77,470 frames. Each frame will be scored by considering several variables, namely hand movement, hand position, and hand blur level in the frame. The combination of these three variables is believed to be able to sort out unimportant frames, namely frames with hands that are transitioning, either transitioning to start the gesture or transitioning to end the gesture. The hand usually moves vertically when starting the sign as well as finishing it. When starting a sign, the hand moves upwards towards the starting position of the sign, and when finishing a sign, the hand will be lowered to return to the starting position, i.e., the ready position. Therefore, hand movements can be used in searching for frames with transitioning hands. To obtain the value of this variable, the calculation involves comparing the center coordinates of the two hands at frame f with those at frame $f + 1$; the highest result from this comparison represents the value of the first variable. The midpoint value is the distance from the minimum and maximum y coordinates divided by two.

Blurriness in the frame is detected by variation of the Laplacian method with the Laplacian kernel used. To get more accurate results, the variation of the Laplacian method is only applied to both hands and not to the entire frame. This is done so the method can really focus on determining the level of blur on hands and not on the entire frame. MediaPipe can find the hand's location, and the axes' min and max coordinates are used to crop it. The method is then applied to the cropped hand image to detect its blurriness level.

The keyframe selection mechanism only focuses on retrieving frames that contain signs without considering the possibility of duplication and redundancy in the frames, so the number of frames is still quite large. The way frame reduction works is by comparing the summation of the coordinates in the head frame with the sum of the coordinates in the frame after the head frame. 'Head frame' is the term used to denote the last frame selected by the system. At the start of the process, the first frame of the sign will be used as the head frame. The summation of the coordinates of the head frame is then compared with the summation of the coordinates of the next frame (in this case, the second frame), and if the difference is smaller than the threshold,

the frame after the head frame will be removed because it is considered similar. If the difference exceeds the threshold, the next frame will be the head frame in the next comparison. The threshold value of a group of frames is obtained by multiplying a positive constant by the average change in the group's keypoint distance.

2.3. Feature extraction: landmark configuration, keypoint extraction, and data augmentation

This process uses MediaPipe holistic to predict landmarks and extract features on the frame. The processes performed are landmark configuration, keypoint extraction, data augmentation, keypoint normalization, and keypoint saving. By default, holistic draws a total of 543 landmarks, consisting of 33 landmarks from pose [25], 21 landmarks (per hand) from hands [26], and 468 landmarks from face mesh [27]. However, in this research, not all landmarks are needed, so some landmarks will not be drawn. There were no changes to hand landmarks, while face mesh landmarks were not used entirely [28]. Pose originally drew 33 landmarks, after being configured pose only drew 17 landmarks. After being configured, pose only draws landmarks on the face area as well as landmarks of both shoulders, elbows, and wrists. The landmarks of pose that are not used are the landmarks of the hand, hip, knee, and foot areas. The total number of landmarks after configuration is 59, which consists of 17 pose landmarks and 21 hand landmarks on each hand.

The MediaPipe coordinate system starts from the top left corner of the frame and is then normalized into a distance of 0 to 1, respectively by the width and height of the image. Each landmark consists of three coordinates, x , y , and z , but in this research only x and y coordinates are extracted for use. Since only two coordinates are extracted, this means that in one frame there are a total of 118 keypoints that can be extracted. The number 118 is derived from the 17 pose landmarks and the 21 hand landmarks for each hand, multiplied by 2, which represents the number of coordinates extracted. Hands that do not appear in the frame will be replaced with a value of 0.

To minimize the possibility of overfitting and improve the generalization of the model, spatial data augmentation of the coordinates was performed. The augmentation parameters were chosen randomly but remained consistent for frames in the same sign group. The augmentation techniques performed on the data are as follows.

2.3.1. Flip

The flip is performed to simulate gestures made with the opposite hand from the active hand in the initial data. The reason for the flip is that in sign language, gestures can be made with either the right or left hand. Gestures made with either hand have the same meaning. Therefore, a horizontal flip is performed so that variations in gestures using the right or left hand can be recognized by the system. Coordinate data that has undergone flip augmentation will also undergo another augmentation process.

2.3.2. Rotation

All coordinates are rotated with a randomly selected angle θ between 6 to 15 degrees, with the center of rotation located at the center of the frame, which is the coordinates $[0.5, 0.5]$. In (1) is a calculation to obtain the radian value from a randomly selected angle, and (2) describes the calculation to obtain the coordinate value after going through the rotation process.

$$\theta = d \cdot \frac{\pi}{180} \quad (1)$$

$$\begin{aligned} \text{rotate}(x, y) = & (x - 0.5) \cos \theta - (y - 0.5) \sin \theta + 0.5, \\ & (y - 0.5) \cos \theta + (x - 0.5) \sin \theta + 0.5 \end{aligned} \quad (2)$$

Where x, y are the input coordinates, x_o, y_o are the coordinates of the center of rotation, d is the degree of angle, θ is the angle of radian.

2.3.3. Squeeze

Squeeze is performed on the x axis with two random ratios r_1 and r_2 between 8 to 15% of the image width w . The y coordinate value is fixed because squeeze is only performed on the horizontal axis. Squeeze makes the proportions of the actor's body smaller, so it can simulate video recording with the actor further away from the camera.

2.3.4. Stretch

Similar to squeeze, stretch is also performed on the horizontal axis with two different random ratios between 8 to 15% of the image width. The y coordinate value remains unchanged. Stretch simulates video recording with the distance of the model closer to the camera by enlarging proportion of the model's body.

2.3.5. Perspective transform

The pose coordinates are projected into a spatially new plane with a randomly defined projection center at an interval between 0 and 1. The new plane is drawn by subtracting the width at the selected side and the vertical edge of each adjacent corner. This augmentation simulates a gesture video recording with a slight camera tilt. The pose coordinates are projected onto a new plane spatially with a randomly determined projection center between 0 and 1. The new plane is drawn by subtracting the width of the selected side and the vertical edge of each adjacent corner.

2.3.6. Arm joint rotation

The coordinates of the arm joints, which include the shoulder, elbow, and wrist, are rotated based on the corresponding joints. It should be noted that this augmentation is only performed on the arm coordinates and not on the palm coordinates, which means that the shape and position of the palm are not changed. Augmentation is only performed on the hand that is active in the signing, which is indicated by the wrist being visible in the frame or not in the ready position. The rotation angle ranges from 2 to 4 degrees, with the chance of each joint rotating being 3:10. The chance of rotating will be increased in the next turn if no rotation occurs in the current turn. The amount of data after going through the data augmentation process is 32,400 videos, or 1,200 per sign. An example of the implementation results of each data augmentation process on one frame is shown in Figure 3. In Figure 3, there are two landmarks for easy observation. The black-colored landmark is the landmark detected by holistic or the initial landmark before going through the augmentation process, while the red-colored landmark is the landmark after going through the augmentation process.

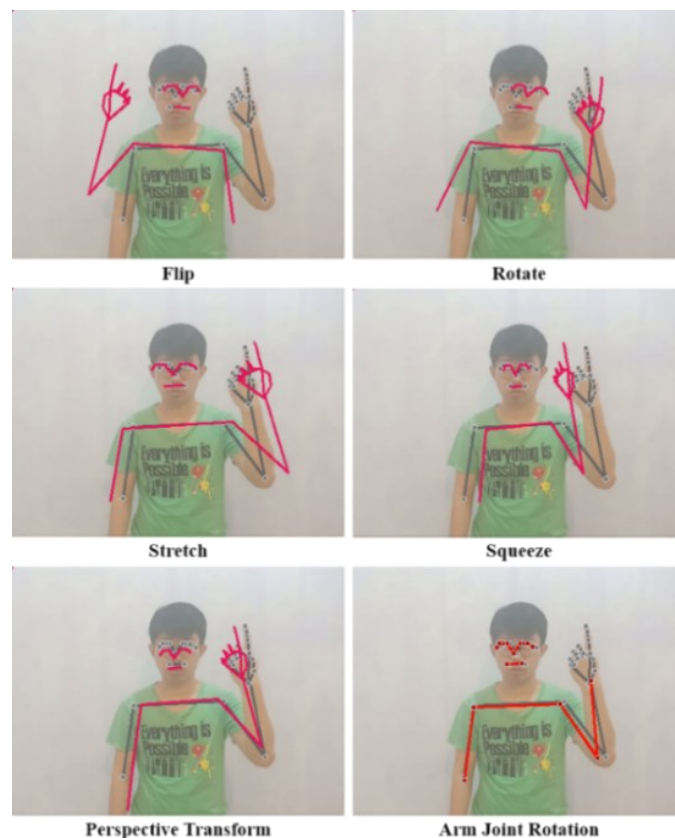


Figure 3. Data augmentation applied on frame

2.4. Keypoint normalization

Keypoint coordinates have values relative to the frame, which makes positional properties such as the position of the performer, distance from the camera, and body proportions of the signer affect these values. Without normalization, the model may learn many spatial features that are not related to the sign itself. Keypoint normalization is performed by restricting the landmarks to their respective bounding boxes of

a predetermined size. The size of the bounding box is determined by applying the "eight-head theory", which states that the height of a human is eight times the height of his head. This normalization method can be applied regardless of the body structure of the performer and the distance of the performer from the camera. The head height is defined as the head unit, and its value is obtained by halving the distance between the signer's shoulders. Normalization of the pose landmark is done by providing a bounding box with a size that covers the entire signing space. Signing space is the space used by the signer during sign language. Most literature [29], [30] mentions that the signing space vertically starts from the waist area to the area slightly above the signer's head and horizontally extends from elbow to elbow when both arms are bent. In this research, the signing space is 6 head units wide with the nose coordinate as its horizontal center or 3 head units to either horizontal side of the nose. The vertical position of the space has a height of 4.5 head units and uses the nose coordinate as its center as well, with the top edge of the space being 0.5 head units away from it and the bottom edge being 4 head units below it. The landmark's coordinates will be based on the bounding box's position.

The hand landmarks are normalized to a bounding box with the width and height of 1.5 head units, with the middle position of the hand as the center. The coordinate value of the hand center position is obtained by halving the distance between the minimum and maximum of the x and y coordinates. Normalization allows the model to actually learn the shape of the hand, rather than just learning the spatial location of the hand within the entire frame. However, the hand location information in the frame is also important, and the model can still learn the hand location information from the Pose landmarks that have both wrist landmarks. The landmark coordinates that have been normalized into their respective bounding boxes are then normalized again into a distance of 0 to 1. An example of applying the keypoint normalization bounding box to one of the frames can be seen in Figure 4.

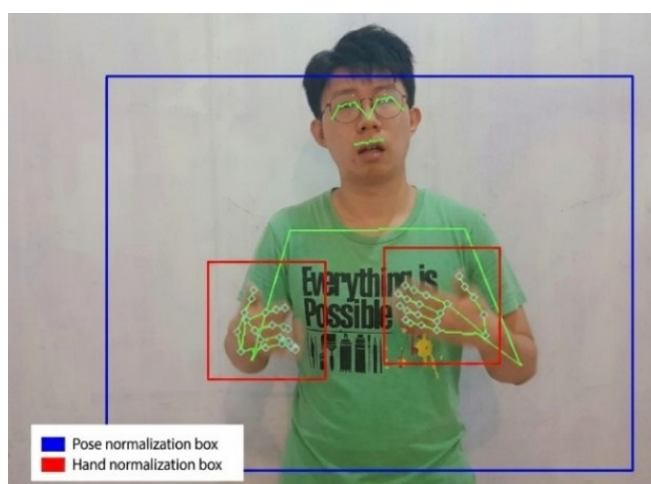


Figure 4. Visualized normalization applied to a frame

2.5. Data preprocessing and experimental setup

The keypoints that have been extracted from each frame will go through preprocessing before they can be used as input for the model to be trained. Data preprocessing includes data padding, data labeling, data splitting, and creating feature and label arrays. Firstly, data padding is a must, because the data series used do not have the same data size. This is due to each sign having a different number of frames. With data padding, all data will have the same data size. Data padding is done by adding a NumPy file containing an array of size (118) with a special symbol 0.1 into each sign directory until it reaches the maximum size, which is 44. The array size of 118 is because it is the number of keypoints extracted in one frame (in one sequence). Data padding will be done post-padding, which is the addition of padding at the end of the data and the maximum size value is the size of the sign data that has the longest data sequence. Second step is data labeling; data labeling is the process of labeling data so that it can be used as a sign for each type of data. Data labeling is done by giving an identity in the form of a number to each sign directory and storing it in a dictionary. The label number starts from 0 and is assigned to each directory that has been sorted alphabetically from small to large.

Then we go through data splitting, data splitting was done with a split ratio of 70:30 (training data : validation data). Data selection during split data is not completely random, but has been set so that each part of the data has the same number of hand variations. This is done so that both hand variations can be recognized by the system properly. The frames that have been randomly selected are put into an array

that holds each data split. The array is then randomized in order, before being written into a .txt file. Lastly, the split data will be put together into two pairs of arrays that hold the features and labels of each data partition. This means that each data partition has two arrays, one array contains data features and one array contains data labels. These arrays will be used as input when training the model. Before continuing to present the model training and testing results, the specification of software and hardware used in this study are as follows: Python 3.9.10, Visual Studio Code v1.72.0, Jupyter Notebook Visual Studio Code, Google Colab, DroidCam v6.5.2, MediaPipe v0.8.11, Tensorflow-GPU v2.10.0, OpenCV-python v4.5.3.56, gTTS v2.2.4, NumPy 1.20.3, and Playsound v1.2.2. While for hardware we were using several laptop and PC with NVIDIA GeForce GTX 1050, AMD Radeon Vega 8 Graphics, and also an Android camera from handphone with DroidCam v6.19 inside.

3. RESULTS AND DISCUSSION

3.1. Training result

BiLSTM hyperparameter tuning is performed with the aim of knowing the best hyperparameter combination so that the performance of the model can be optimized. Hyperparameter tuning focuses on the hyperparameters of the number of neuron units of the BiLSTM layer, the batch size value, and the value of the dropout layer. The data used as input for the training process is four data arrays, two as training data and two as validation data. Each data point has one array holding the features of the data and one array holding the label of the data. In the training process, we utilize the Google Colab platform with a GPU because it is considered to have faster computation compared to our device. The training process is carried out for 5 epochs, and the checkpoint with the highest accuracy is taken.

There are three combinations of the number of neuron units tested, those combinations are 64, 128, 64; 128, 256, 128; and 256, 512, 256. The first number represents the number of neurons in the first BiLSTM layer, the second number represents the number of neurons in the second layer, and the third number represents the number of neurons in the last layer. For batch size, tested values are 32, 64, and 128. For the dropout value, the values to be tested are 0.2 and 0.5. The model is trained with all hyperparameter combinations, which means eighteen training trials will be conducted. The validation accuracy value and the number of epochs is used as a reference in determining the best hyperparameter combination. Table 1 is the result of hyperparameter tuning; the epoch column displays the epoch value when the model reaches the best validation accuracy.

Table 1. BiLSTM hyperparameter tuning results

Number of neuron units	Batch size	Dropout	Accuracy (%)	Epoch
(64, 128, 64)	32	0.2	95.07	5
(64, 128, 64)	32	0.5	77.05	5
(64, 128, 64)	64	0.2	93.03	5
(64, 128, 64)	64	0.5	73.01	2
(64, 128, 64)	128	0.2	92.09	4
(64, 128, 64)	128	0.5	77.47	5
(128, 256, 128)	32	0.2	94.8	4
(128, 256, 128)	32	0.5	90.31	4
(128, 256, 128)	64	0.2	92.63	4
(128, 256, 128)	64	0.5	86.52	5
(128, 256, 128)	128	0.2	85.77	5
(128, 256, 128)	128	0.5	84.77	5
(256, 512, 256)	32	0.2	97.48	5
(256, 512, 256)	32	0.5	88.25	5
(256, 512, 256)	64	0.2	93.23	5
(256, 512, 256)	64	0.5	93.12	4
(256, 512, 256)	128	0.2	95.99	5
(256, 512, 256)	128	0.5	91.89	5

Based on Table 1, it is known that the model with a combination of the number of neurons (256, 512, 256) with a batch size of 32 and a dropout value of 0.2 has the highest accuracy value of 97.48% in 5 epochs. Thus, this hyperparameter configuration will be used. Figure 5 shows the accuracy and loss graph of the BiLSTM model, while Figure 6 shows the accuracy and loss graph for the best TCN model. For the TCN, separate training was conducted from the BiLSTM model. At this stage, TCN hyperparameter tuning was performed using Keras tuner to find the best hyperparameters for building, optimizing, and training the TCN model. The parameters tested were the number of layers, filter units, activation, dilation rate, dropout rate, dense unit, and batch size. Testing was carried out 30 times, with each experiment having a

maximum of 10 epochs. The best parameter results and details of each experiment can be seen in Table 2. After hyperparameter tuning, the highest accuracy was achieved with the combination of neurons (64, 64, 128), a dilation of 1, 4, and 8, and a dropout rate of 0.4; therefore, this combination was implemented in the model.

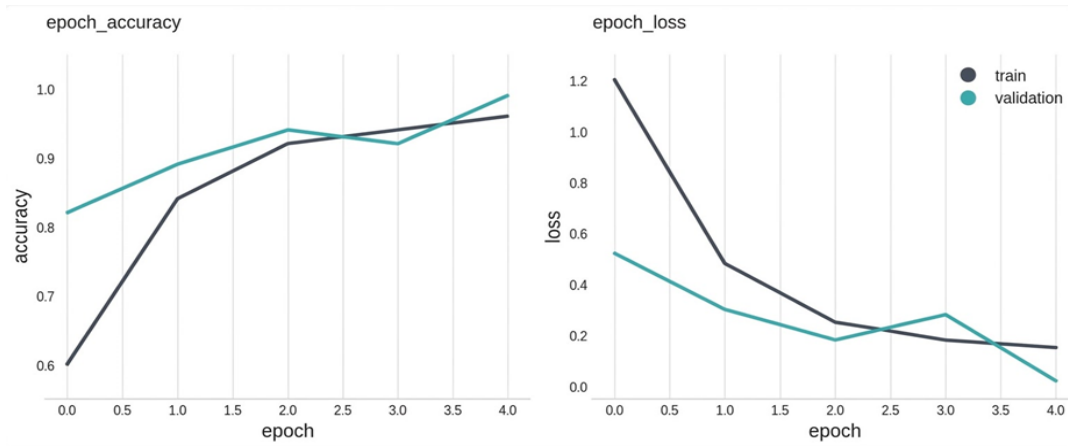


Figure 5. BiLSTM model's accuracy and loss

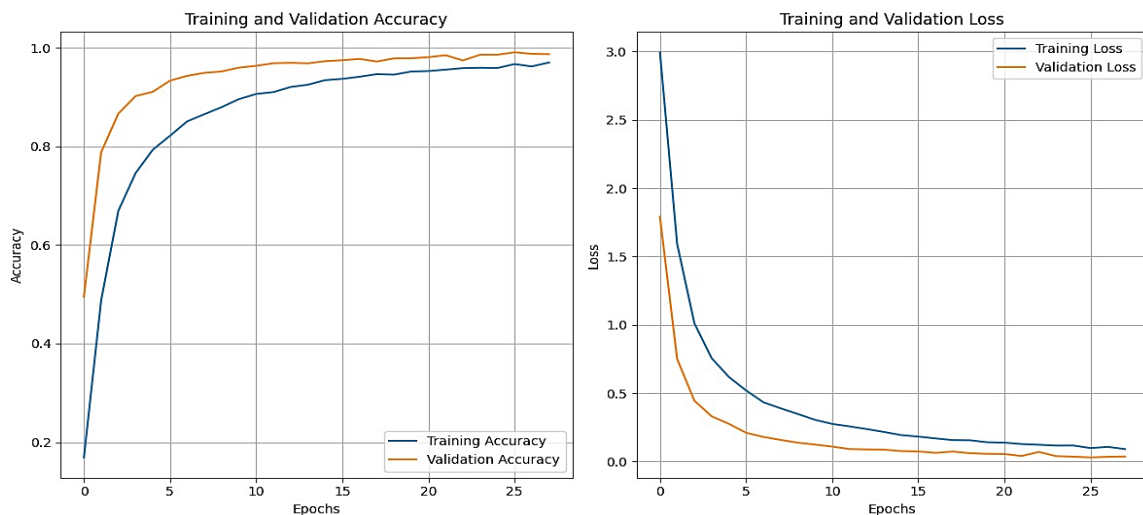


Figure 6. TCN model's accuracy and loss

3.2. Testing result

Model testing is done in real time and each word will be tested 30 times with both hand variations, 15 times for right-hand variations and 15 times for left-hand variations. This means the system will be tested 810 times. The incoming frames will go through a series of preprocessing steps, such as frame resizing, frame selection, keypoint extraction, and keypoint normalization. Table 3 shows all evaluation metric for each model, while Figure 7 shows the confusion matrix comparison between BiLSTM and TCN, all in MediaPipe platform. Where Figure 7(a) shows the confusion matrix for BiLSTM and Figure 7(b) shows the confusion matrix for TCN.

Based on Table 3, it can be seen that out of the 810 data sets used for testing, 698 correct data sets were generated for the BiLSTM model and 718 correct data sets for the TCN. In other words, BiLSTM produced an accuracy of 86.17%, while TCN outperformed BiLSTM with an accuracy of 88.64%. It is also possible to observe that TCN has 92 data points, whereas BiLSTM has 112 data points, with regard to the data that falls into the false negative group. This suggests that the use of BiLSTM as a model result in a greater number of mistakes when compared to the use of TCN. In addition, when we examine the precision values for each label that was applied, we notice that TCN is more dominant than BiLSTM in almost all labels. Recall and F1-score are also more dominant than BiLSTM.

Table 2. TCN hyperparameter tuning results

Layer count	Filter 1	Act 1	Dil 1	Filter 2	Act 2	Dil 2	Filter 2	Act 3	Dil 3	Dense	Dropout	Accuracy (%)
3	64	tanh	1	64	relu	4	128	tanh	8	256	0.4	97.89
3	32	tanh	4	512	relu	1	256	tanh	2	256	0.4	97.87
3	128	tanh	1	64	relu	2	64	tanh	8	256	0.2	97.78
3	32	tanh	2	64	tanh	1	512	tanh	8	256	0.2	97.53
2	128	relu	8	256	tanh	4	-	-	-	512	0.2	97.44
3	64	tanh	4	128	tanh	1	128	relu	4	128	0.3	97.39
3	64	relu	8	256	tanh	1	64	relu	4	256	0.2	96.76
2	128	relu	1	32	relu	4	-	-	-	512	0.2	96.75
2	512	tanh	1	32	tanh	4	-	-	8	512	0.3	96.10
2	128	tanh	8	32	relu	1	-	-	-	256	0.3	96.01
2	64	relu	8	64	tanh	8	-	-	-	128	0.2	95.47
1	512	relu	8	-	-	-	-	-	-	256	0.2	95.44
3	64	tanh	1	32	relu	8	64	tanh	1	128	0.3	95.25
1	256	tanh	8	-	-	-	-	-	-	512	0.2	95.03
1	256	tanh	1	-	-	-	-	-	-	128	0.2	94.97
1	128	relu	2	-	-	-	-	-	-	512	0.4	94.76
1	512	relu	2	-	-	-	-	-	-	256	0.3	94.69
1	64	tanh	8	-	-	-	-	-	-	512	0.4	94.49
3	256	relu	2	32	relu	1	32	relu	1	128	0.3	94.22
1	512	relu	4	-	-	-	-	-	-	128	0.2	94.14
1	64	tanh	8	-	-	-	-	-	-	256	0.3	94.06
1	64	tanh	4	-	-	-	-	-	-	128	0.3	93.93
2	32	tanh	2	32	tanh	8	-	-	-	256	0.4	93.76
2	512	relu	1	32	relu	4	-	-	-	128	0.4	93.69
1	128	tanh	8	-	-	-	-	-	-	128	0.3	93.62
3	128	relu	4	128	tanh	1	32	tanh	2	512	0.3	93.54
1	64	relu	4	-	-	-	-	-	-	512	0.4	93.43
1	32	relu	8	-	-	-	-	-	-	512	0.2	92.51
1	64	relu	2	-	-	-	-	-	-	128	0.2	92.14
1	32	tanh	8	-	-	-	-	-	-	128	0.4	91.67

Table 3. BiLSTM and TCN comparison results

Label	BiLSTM						TCN					
	TP	FN	FP	Precision	Recall	F1-score	TP	FN	FP	Precision	Recall	F1-score
What	26	4	11	0.70	0.87	0.78	26	4	3	0.90	0.87	0.88
How	19	11	4	0.83	0.63	0.72	27	3	4	0.87	0.90	0.88
Sir	29	1	9	0.76	0.97	0.85	30	0	6	0.83	1.00	0.91
How many	23	7	5	0.82	0.77	0.79	25	5	9	0.73	0.83	0.78
Tomorrow	27	3	0	1.00	0.90	0.95	27	3	0	1.00	0.90	0.95
He/She	30	0	3	0.91	1.00	0.95	30	0	5	0.86	1.00	0.92
Day	28	2	14	0.67	0.93	0.78	28	2	9	0.76	0.93	0.84
Today	30	0	3	0.91	1.00	0.95	30	0	3	0.91	1.00	0.95
You	25	5	15	0.62	0.83	0.71	27	3	10	0.73	0.90	0.81
When	23	7	2	0.92	0.77	0.84	21	9	0	1.00	0.70	0.82
Where	21	9	2	0.91	0.70	0.79	22	8	1	0.96	0.73	0.83
Yesterday	28	2	5	0.85	0.93	0.89	27	3	5	0.84	0.90	0.87
Why	24	6	1	0.96	0.80	0.87	24	6	0	1.00	0.80	0.89
We	21	9	0	1.00	0.70	0.82	21	9	0	1.00	0.70	0.82
Sorry	24	6	1	0.96	0.80	0.87	25	5	1	0.96	0.83	0.89
Eat	28	2	5	0.85	0.93	0.89	28	2	6	0.82	0.93	0.88
Name	30	0	0	1.00	1.00	1.00	30	0	0	1.00	1.00	1.00
Number	30	0	0	1.00	1.00	1.00	30	0	0	1.00	1.00	1.00
I	30	0	0	1.00	1.00	1.00	30	0	0	1.00	1.00	1.00
Now	27	3	0	1.00	0.90	0.95	27	3	0	1.00	0.90	0.95
Good night	21	9	0	1.00	0.70	0.82	21	9	2	0.91	0.70	0.80
Good morning	27	3	19	0.59	0.90	0.71	27	3	13	0.68	0.90	0.77
Good afternoon	25	5	4	0.86	0.83	0.85	25	5	4	0.86	0.83	0.85
Who	22	8	0	1.00	0.73	0.85	26	4	0	1.00	0.87	0.93
Phone	30	0	3	0.91	1.00	0.95	30	0	3	0.91	1.00	0.95
Thank you	21	9	6	0.78	0.70	0.74	25	5	8	0.76	0.83	0.80
Deaf	29	1	0	1.00	0.97	0.98	29	1	0	1.00	0.97	0.98
Total	698	112	112	-	-	-	718	92	92			

MediaPipe was also tested at various camera distances, with the tested distances being 50 cm, 100 cm, and 150 cm. MediaPipe's capability for initial detection offers a fundamental difference [17], particularly compared to frameworks that perform initial annotation, such as you only look once (YOLO) [1]. The system's test results for these various distances can be seen in Table 4. MediaPipe was able to detect at all tested distances, but its performance began to decline at 150 cm. At this distance, detection becomes

inconsistent and often mispredicts landmarks, which can affect the frame selection process and classification results. Therefore, it can be concluded that the optimal distance for the best detection by the model is between 50 cm and 100 cm. The selection of this distance is very important in the implementation of real-time systems to ensure that the model accuracy remains high during the BISINDO gesture recognition process.

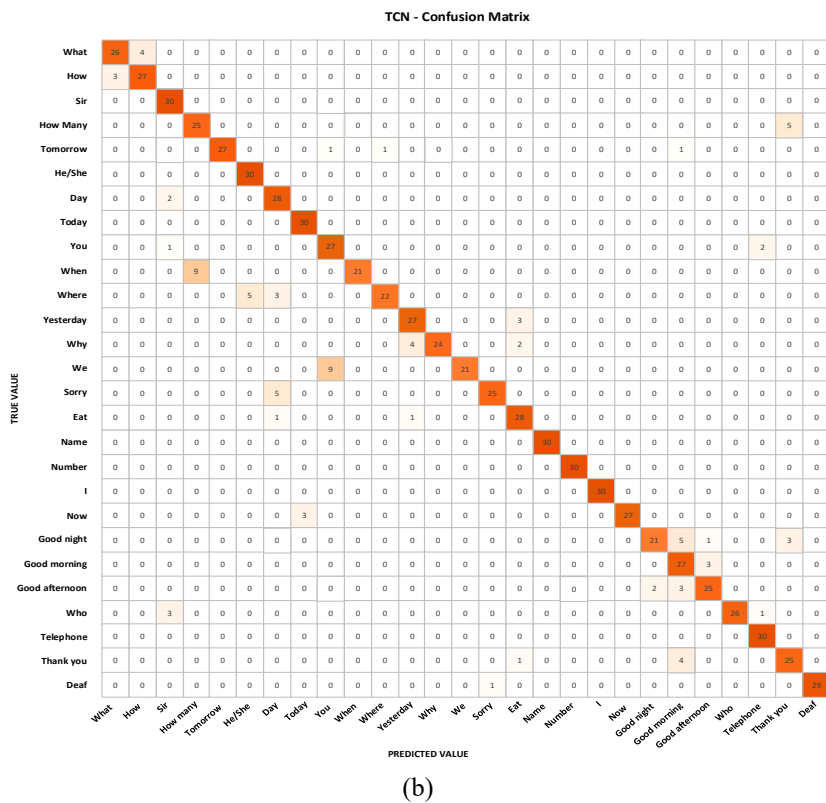
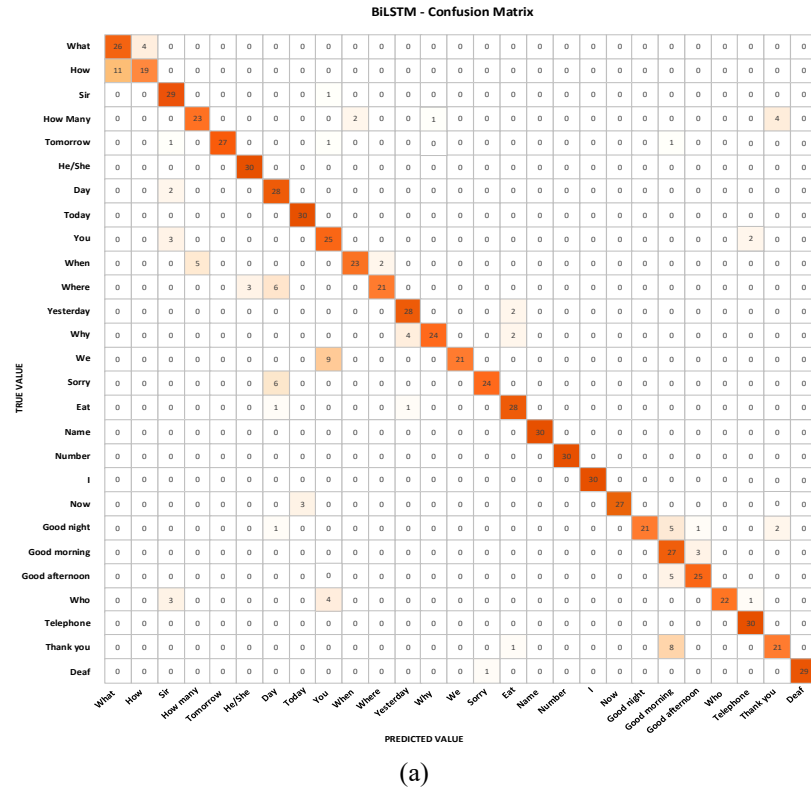


Figure 7. Confusion matrix comparison of (a) BiLSTM and (b) TCN

Based on the test results, the BISINDO gesture recognition model using the TCN algorithm demonstrated an accuracy rate of 88.64% or 2.47% better than BiLSTM. This result reflects the model's ability to classify gestures quite well, although there were still some errors in prediction. One of the main factors contributing to misclassification is the similarity of movements between gestures, especially at the beginning, middle, and end stages. For example, the movements in the what and how classes have similar patterns, so the system had difficulty distinguishing between the two classes, resulting in false negatives and false positives. Similar errors were also found in several other classes with nearly identical movement elements, such as good night and good morning. Furthermore, lighting levels and camera angles during testing can also affect prediction accuracy. These environmental factors can cause variations in the visual representation of each movement, potentially increasing the number of classification errors. Therefore, improving data preprocessing and image augmentation can be an effective strategy to reduce errors caused by varying lighting conditions and angles.

The effect of applying data augmentation and keypoint normalization on model performance was also analyzed by training three models: A, B, and C. Each model is tested on the same test data so that their performance can be compared. Models will only be trained for 30 epochs or stopped when the validation accuracy has reached 90%. Model A is trained with data that has been augmented but not through the keypoint normalization process. In contrast, model B is trained with data that has only gone through flip augmentation and has gone through the keypoint normalization process. Model C is trained using data that has been augmented and normalized. Table 5 shows the results of the ablation study on the test data.

Model A only gets an accuracy value of 33.33% for BiLSTM and 35.67% for TCN, which shows that keypoint normalization greatly affects the performance of the model. This is evident in the other two models that apply keypoint normalization and obtain accuracy values that are far above model A. In model B, TCN outperformed BiLSTM, achieving an accuracy of 64.56%, compared to BiLSTM's 60.86%. Thus, when the implementation of data augmentation and normalization implemented, then the improvement become significant in model C, the increment is around 25.31% for BiLSTM and 24.08% for TCN. The results of this ablation study show that applying data augmentation as well as keypoint normalization can improve the performance of the model significantly.

The presented test results indicate the presence of some translation errors in the model. This study analyzes and concludes that there are several possibilities that can cause classification errors by the model, including i) less than optimal keyframe selection results. The classification process relies heavily on the results of keyframe selection, so improperly selected frames can lead to errors during the classification process. The number of frames left after the selection process can also affect the classification results; ii) errors in MediaPipe holistic, such as position estimation errors or the inability to estimate landmarks, can significantly impact the classification results. MediaPipe has weaknesses when detecting side and front views of hands; and iii) similarity factors in the signs, such as hand shape and hand gesture patterns, can affect classification results.

Table 4. MediaPipe distance testing

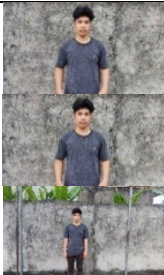
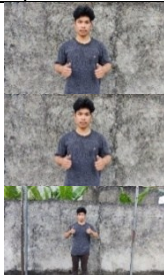
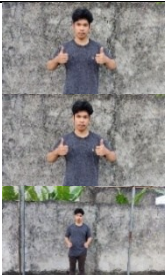
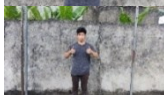
No.	Distance (cm)	Image capture based on distance			Description
1	50				MediaPipe is capable of accurate detection.
2	100				MediaPipe is capable of accurate detection.
3	150				MediaPipe cannot detect accurately.

Table 5. Ablation study results

Model	Augmentation	Normalization	Accuracy of BiLSTM (%)	Accuracy of TCN (%)
A	✓	✗	33.33	35.67
B	✗	✓	60.86	64.56
C	✓	✓	86.17	88.64

4. CONCLUSION

In this research, a system for translating BISINDO was successfully designed and implemented using a combination of BiLSTM + MediaPipe, and TCN + MediaPipe methods. The augmentation and

normalization techniques used in this study are effective in improving the generalization and accuracy of the model. BISINDO translation using the combination model of BiLSTM and MediaPipe method gets an accuracy as high as 86.17% on 810 test data. While for TCN and MediaPipe get the best result in 88,64% on the same dataset. Detection errors in the system are caused by several factors, such as suboptimal frame selection results, errors in MediaPipe holistic, and similarity factors in some signs. Misclassification occurred in several gestures with high movement similarity, particularly in the what-how, and good night-good morning classes. MediaPipe is able to detect signals accurately at distances of 50 cm to 100 cm, but detection performance decreases at distances of 150 cm. The TCN algorithm is able to handle sequential data well, recognizing hand movement patterns in sign language with high accuracy compared to BiLSTM. In terms of future work, we know that our research can still be continued and expanded as future research in sentence-wise translator systems. This will be the next objective. To improve the accuracy and generalization of the model, it is recommended to use a larger and more diverse dataset, including variations in hand movements, movement speeds, and lighting conditions. Integration with more advanced cameras can help improve system performance in real-world conditions.

FUNDING INFORMATION

This research is supported by Universitas Sumatera Utara under TALENTA research grant with number: 30/UN5.4.10.S/PPM/KP-TALENTA/RB1/2024.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sarah Purnamawati	✓			✓		✓			✓	✓	✓	✓		✓
Albert	✓	✓	✓		✓		✓		✓					
Anggi Yohanes Pardede	✓	✓	✓		✓		✓		✓					
Iqbal Fakhri	✓	✓	✓		✓		✓			✓	✓		✓	
Farhad Nadi	✓	✓		✓	✓			✓		✓		✓		
Mohammad Fadly	✓			✓	✓			✓	✓			✓	✓	
Syahputra														
Romi Fadillah Rahmat	✓	✓			✓	✓			✓	✓	✓	✓		✓

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nterpretation

R : **R**esources

D : **D**ata Curation

O : **O**riginal Draft

E : **E**diting

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest regarding political, personal, religious, ideological, academic, intellectual competing interests, competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

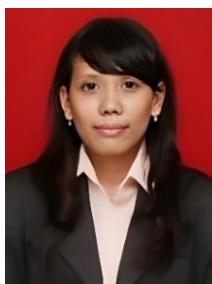
The data that support the findings of this study are available from the corresponding author, [SP], upon reasonable request.

REFERENCES

- [1] S. Daniels, N. Suciati, and C. Fathichah, "Indonesian sign language recognition using YOLO method," *IOP Conference Series: Materials Science and Engineering*, vol. 1077, no. 1, Feb. 2021, doi: 10.1088/1757-899X/1077/1/012029.
- [2] B. Alsharif, A. S. Altaher, A. Altaher, M. Ilyas, and E. Alalwany, "Deep learning technology to recognize American sign language alphabet," *Sensors*, vol. 23, no. 18, Sep. 2023, doi: 10.3390/s23187970.
- [3] S. Albanie *et al.*, "BSL-1K: scaling up co-articulated sign language recognition using mouthing cues," in *Computer Vision – ECCV 2020*, Cham, Switzerland: Springer, 2020, pp. 35–53, doi: 10.1007/978-3-030-58621-8_3.

- [4] S. Kumar, R. Rani, S. K. Pippal, and U. Chaudhari, "Real time Indian sign language recognition using transfer learning with VGG16," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 22, no. 6, pp. 1459–1468, Dec. 2024, doi: 10.12928/telkomnika.v22i6.26498.
- [5] S. Isnaniah, T. Agustina, Islahuddin, and F. Annisa, "The use of sign language in deaf Indonesian classrooms in Surakarta," *KEMBARA Journal of Scientific Language Literature and Teaching*, vol. 9, no. 2, pp. 468–481, Oct. 2023, doi: 10.22219/kembara.v9i2.25990.
- [6] S. Dwijayanti, Hermawati, S. I. Taqiyyah, H. Hikmarika, and B. Y. Suprpto, "Indonesia sign language recognition using convolutional neural network," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, pp. 415–422, 2021, doi: 10.14569/IJACSA.2021.0121046.
- [7] N. Palfreyman, "Sign language varieties of Indonesia," *Sign Language & Linguistics*, vol. 20, no. 1, pp. 135–145, Nov. 2017, doi: 10.1075/sll.20.1.06pal.
- [8] D. Joan, V. Vincent, K. J. Daniel, S. Achmad and R. Sutoyo, "BISINDO hand-sign detection using transfer learning," *2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, Kuala Lumpur, Malaysia, 2023, pp. 1-7, doi: 10.1109/ICRAIE59459.2023.10468194.
- [9] S. R. Kodandaram, N. P. Kumar, and G. L. Sunil, "Sign language recognition," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 14, pp. 994–1009, 2021, doi: 10.17762/turcomat.v12i14.10381.
- [10] D. K. Singh, "3D-CNN based dynamic gesture recognition for Indian sign language modeling," *Procedia Computer Science*, vol. 189, pp. 76–83, 2021, doi: 10.1016/j.procs.2021.05.071.
- [11] M. D. Coster, M. V. Herreweghe, and J. Dambre, "Sign language recognition with transformer networks," in *LREC 2020 - 12th International Conference on Language Resources and Evaluation*, 2020, pp. 6018–6024.
- [12] A. Tunga, S. V. Nuthalapati, and J. Wachs, "Pose-based sign language recognition using GCN and BERT," in *2021 IEEE Winter Conference on Applications of Computer Vision Workshops*, Jan. 2021, pp. 31–40, doi: 10.1109/WACVW52041.2021.00008.
- [13] I. A. Adeyanju, O. O. Bello, and M. A. Adegboye, "Machine learning methods for sign language recognition: a critical review and analysis," *Intelligent Systems with Applications*, vol. 12, Nov. 2021, doi: 10.1016/j.iswa.2021.200056.
- [14] Z. Katılmış and C. Karakuzu, "ELM-based two-handed dynamic Turkish sign language (TSL) word recognition," *Expert Systems with Applications*, vol. 182, Nov. 2021, doi: 10.1016/j.eswa.2021.115213.
- [15] S. Ghosh, "Proposal of a real-time American sign language detector using MediaPipe and recurrent neural network," *International Journal of Computer Sciences and Engineering*, vol. 9, no. 7, pp. 46–52, Jul. 2021, doi: 10.26438/ijcse/v9i7.4652.
- [16] A. Aljabar and S. Suharjito, "BISINDO (Bahasa Isyarat Indonesia) sign language recognition using CNN and LSTM," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 5, pp. 282–287, 2020, doi: 10.25046/aj050535.
- [17] A. Halder and A. Tayade, "Real-time vernacular sign language recognition using MediaPipe and machine learning," *International Journal of Research Publication and Reviews*, vol. 2, no. 5, pp. 9–17, 2021.
- [18] P. Tsinganos, B. Jansen, J. Cornelis, and A. Skodras, "Real-time analysis of hand gesture recognition with temporal convolutional networks," *Sensors*, vol. 22, no. 5, Feb. 2022, doi: 10.3390/s22051694.
- [19] M. Ishaq, M. Khan, and S. Kwon, "TC-Net: a modest & lightweight emotion recognition system using temporal convolution network," *Computer Systems Science and Engineering*, vol. 46, no. 3, pp. 3355–3369, 2023, doi: 10.32604/csse.2023.037373.
- [20] Z. Katılmış and C. Karakuzu, "Double-handed dynamic Turkish sign language recognition using Leap Motion with meta-learning approach," *Expert Systems with Applications*, vol. 228, Oct. 2023, doi: 10.1016/j.eswa.2023.120453.
- [21] S. Siddique, S. Islam, E. E. Neon, T. Sabbir, I. T. Naheen, and R. Khan, "Deep learning-based Bangla sign language detection with an edge device," *Intelligent Systems with Applications*, vol. 18, May 2023, doi: 10.1016/j.iswa.2023.200224.
- [22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997, doi: 10.1109/78.650093.
- [23] M. Yang and J. Wang, "Adaptability of financial time series prediction based on BiLSTM," *Procedia Computer Science*, vol. 199, pp. 18–25, 2022, doi: 10.1016/j.procs.2022.01.003.
- [24] C. Lugaresi *et al.*, "MediaPipe: a framework for building perception pipelines," in *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)*, 2019, pp. 1–4.
- [25] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "BlazePose: on-device real-time body pose tracking," 2020, *arXiv:2006.10204*.
- [26] H.-H. Li and C.-C. Hsieh, "Dynamic hand gesture recognition using MediaPipe and transformer," *Engineering Proceedings*, vol. 108, no. 1, Sep. 2025, doi: 10.3390/engproc2025108022.
- [27] S. Hangaragi, T. Singh, and N. N., "Face detection and recognition using face mesh and deep neural network," *Procedia Computer Science*, vol. 218, pp. 741–749, 2023, doi: 10.1016/j.procs.2023.01.054.
- [28] M. Al-Hammadi *et al.*, "Spatial attention-based 3D graph convolutional neural network for sign language recognition," *Sensors*, vol. 22, no. 12, Jun. 2022, doi: 10.3390/s22124558.
- [29] B. Lenseigne and P. Dalle, "Using signing space as a representation for sign language processing," in *Gesture in Human-Computer Interaction and Simulation (GW 2005)*, Berlin, Germany: Springer, 2006, pp. 25–36, doi: 10.1007/11678816_3.
- [30] A. Bauer, *The use of signing space in a shared sign language of Australia*. Berlin, Germany: Walter de Gruyter, 2014, doi: 10.1515/9781614515470.

BIOGRAPHIES OF AUTHORS



Sarah Purnamawati    is a researcher and lecturer at Universitas Sumatera Utara. Currently, she is a vice dean in the Faculty of Computer Science and Information Technology. She finished her master's program at Universiti Sains Malaysia. Her research interests are in database and image processing. She can be contacted at email: sarah_purnamawati@usu.ac.id.



Albert     was a student and assistant researcher at Universitas Sumatera Utara. He currently works at Wilmar Consultancy Services as a web developer. His expertise lies in web and mobile development, and successfully worked on a variety of projects using Java, Kotlin, Python, PHP, JavaScript, and SQL. He can be contacted at email: albertaries18@gmail.com.



Anggi Yohanes Pardede     was graduated from his undergraduate studies from the Faculty of Computer Science and Information Technology, Universitas Sumatera Utara. He currently works as a full-stack developer for several projects in a government institution in Indonesia. He can be contacted at email: anggiyohanespdd@gmail.com.



Iqbal Fakhri     was a student and assistant researcher at Universitas Sumatera Utara. Currently, he is a full-stack developer at PT. Lion Superindo. He finished his undergraduate from Universitas Sumatera Utara in 2024. His research interests are in prediction optimization, deep learning, and information technology. He can be contacted at email: iqbalfakhri@usu.ac.id.



Farhad Nadi     holds a Ph.D. in Computer Science from Universiti Sains Malaysia and has extensive experience in both industry and teaching. Currently, he is an associate professor at the School of Information Technology, UNITAR International University, Malaysia. He can be contacted at email: farhad.nadi@unitar.my.



Mohammad Fadly Syahputra     is a researcher and lecturer at Universitas Sumatera Utara. He received his master's from Universiti Utara Malaysia and is currently pursuing his Ph.D. at Universitas Sumatera Utara. His research interests are in computer learning environments, virtual reality, and pedagogical agents. He can be contacted at email: nca.fadly@usu.ac.id.



Romi Fadillah Rahmat     is a professor, researcher, and lecturer at Universitas Sumatera Utara. Currently, he is a vice dean in the Faculty of Computer Science and Information Technology at Universitas Sumatera Utara. He finished his undergraduate and master's programs at Universiti Sains Malaysia and completed his Ph.D. at Universitas Sumatera Utara. His research interests are in image processing, deep learning, and mobile computing. He can be contacted at email: romi.fadillah@usu.ac.id.