Artificial Intelligence-Based System for Word Recognition and Extraction in Indonesian Sign Language Videos

1st Marvel Martawidjaja
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
marvel.martawidjaja@binus.ac.id

2nd Richie Darylson Tandiono
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
richie.tandiono@binus.ac.id

3rd Lili Ayu Wulandhari
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
lili.wulandhari@binus.ac.id

***Abstract*—The Indonesian Sign Language System (SIBI) is a standardized sign language for deaf and mute communities, particularly in formal education. However, current solutions usually use static images or translate only alphabets. Furthermore, studies involving words used in academic settings are limited. This study develops and evaluates a 3D Convolutional Neural Network (3D CNN) and a CNN-LSTM model to recognize SIBI gestures commonly used in academic settings from recorded video data. Data preprocessing was done using frame resizing, temporal sampling, and data augmentation techniques to enhance its variability and robustness. Experimental results show that the CNN-LSTM model consistently outperforms the 3D CNN, achieving accuracies and F1 scores above 0.95 across all dataset variations. However, the CNN-LSTM has slower inference (2.70 ms per frame) compared to the 3D CNN (0.74 ms per frame). On top of that, a real-time SIBI gesture recognition application is made using the CNN-LSTM model to evaluate its performance in real-life settings on different hardware configurations. These findings highlight the trade-off between accuracy and computational efficiency, suggesting CNN-LSTM is more suitable for accuracy-critical applications, while 3D CNN offers advantages for real-time deployment. Future research should expand the dataset scope and explore advanced model architectures to further improve recognition of SIBI words.**

***Index Terms*—sign language recognition, convolutional neural networks, long short-term memory, video classification, deep learning**

I. Introduction

The Indonesian Sign Language System (SIBI) is a sign language created in Indonesia to facilitate communication among deaf and mute groups, especially within formal educational contexts. SIBI was initially conceived in the 1970s and was subsequently introduced in 1994 by the Indonesian Ministry of Education. As a sign language, SIBI is derived from the American Sign Language (ASL) and resembles the formal Indonesian language in several ways, such as having gestures for affixes [2]. However, many people prefer to use Indonesian Sign Language (BISINDO). The main reason for this is the simplicity of BISINDO and also because many people are already used to BISINDO as a sign language for their day-to-day lives [2, 3].

The limitations in technologies to translate SIBI gestures into text or audio limit accessibility for the deaf and mute communities. Artificial Intelligence (AI) and Computer Vision have the potential to address these challenges. AI systems have shown that they can process video data in real-time, recognize sign language gestures, and translate them into text to improve communication for the hearing impaired [4]. Many studies have used AI applications for BISINDO, such as F. A. Setiawan et al., who utilized You Only Look Once (YOLO) with CNN for BISINDO alphabet detection [5]; R. Setiawan et al., who applied CNN and Long Short-Term Memory (LSTM) for BISINDO recognition [6]; and M. Z. Fauzi et al., who created real-time BISINDO translation [7]. Meanwhile, SIBI-specific deep learning applications remain under-explored. Limantara and Tristianto employed a CNN model to classify SIBI alphabets on images [8]. Other works use SIBI gesture videos, like R. A. T. Kurniawan & W. Kaswidjanti, who used videos of SIBI question words to train an LSTM model [9]. Nonetheless, research on a broader set of SIBI gestures, especially in formal education settings, are still few. Therefore, this research aims to develop AI-based models for the recognition of SIBI gestures from videos. The developed models utilize computer vision techniques and deep learning to improve communication and accessibility for Indonesia's deaf community, particularly in formal education contexts.

This paper consists of five sections: Section I contains the introduction; Section II contains the review of existing literature on sign language recognition; Section III contains the methodology used in this paper; Section IV contains the results obtained and its discussion; Section V contains the conclusion of this paper.

II. Literature Review

A. Overview of Sign Language Recognition (SLR)

There are many studies that developed deep learning models combined with implementations of computer vision principles for Sign Language Recognition (SLR) tasks. J. Huang & V. Chouvatut developed a ResNet-based CNN followed by an LSTM layer. They successfully achieved an accuracy of over 86% [10]. Camgoz et al. proposed a transformer-based model for continuous sign language translation [11]. Many input

modalities have also been explored. R. Cui et al. combined RGB images with their optical flow in order to capture motion and spatial information [12]. N. Kolotouros et al. on the other hand, used a keypoint-based method by extracting hand and body landmarks [13].

B. Sign Language Recognition in Indonesia

Many research done on sign language recognition in Indonesia use BISINDO as their input data. On the contrary, research using SIBI data are fewer in quantity. Limantara and Tristianto [8] used a CNN model to classify static SIBI alphabet gestures. A. N. Handayani et al. [14] compared CNN models like ResNet-50 and EfficientNet for SIBI alphabet recognition, using keypoint-based inputs. Their EfficientNet model reached an accuracy of 99.1%, showing that CNN models can detect SIBI alphabet gestures with a high degree of accuracy. However, challenges remain present in using these models to handle video inputs and continuous signing.

C. Related Works

Deep learning-based models are popular options for a wide variety of tasks, including SLR. The architecture of deep learning models contains several interconnected layers that have a large number of learnable parameters. Empirical results show that deep learning-based models can achieve good accuracy in various sign languages [15].

A convolutional neural network (CNN) is a type of neural network that contains convolution layers, which use a kernel or a small sliding window to perform a convolution operation on the input. Usually, a CNN architecture that is used to encode features of an image uses a 2-by-2 kernel for convolution. To encode the additional temporal dimension that is present in a video, a 3D CNN architecture can be used. 3D CNN is a CNN architecture that uses a 3-by-3 kernel to perform convolution. Much research has been done on SLR using 3D CNN. D. K. Singh used a 3D CNN to perform SLR on an Indian Sign Language (ISL) with noisy backgrounds and achieved 88.52 an F1 score [16]. This shows that a 3D CNN architecture can be effective in SLR tasks. However, as mentioned in [17], a 3D-CNN architecture can face overfitting problems and have difficulty with fast movements.

Recurrent Neural Network (RNN) is a type of model architecture that is designed to handle sequential data, where the order of the inputs are important, such as texts or videos. This is done by using recursive blocks that receive the output of the block from the previous time step while also receiving input at the current time step. Long Short-Term Memory (LSTM) is a type of RNN that can retain long-term dependencies in sequential data, making it suitable for longer sequences of data. In order to process video data, a CNN-LSTM hybrid can be used, where the CNN encodes spatial features from each frame and the LSTM encodes sequential dependencies between the CNN results. J. Huang & V. Chouvatut used a CNN-LSTM hybrid, where the CNN portion of the architecture used a pretrained ResNet. The model obtained an accuracy of 86.25% on the Argentine Sign Language dataset (LSA64) [10]. The results suggest that CNN-LSTM architecture may be suitable for SIBI SLR task. The limitations of this architecture are the low convergence rate and low sensitivity to similar data, as mentioned by [18].

III. Methodology

The experimentation process was carried out in six main steps, as illustrated in Fig. 1 The first step was data collection, where a dataset of 20 SIBI gestures was gathered from 28 participants and validated for consistency. The second step was data processing, which involved resizing and normalizing the frames as well as applying temporal sampling. Various parameter settings in these preprocessing techniques were tested to evaluate their effect on model performance. In the third step, dataset variants were created, consisting of the original dataset and two augmented versions. Augment-1 has geometric transformations, while Augment-2 added temporal sampling variations on top of Augment-1. The fourth step was data splitting, where the dataset was divided into training, validation, and testing sets with an 8:1:1 ratio. After that, two different deep learning model architectures were trained: a 3D CNN and a CNN-LSTM. The 3D CNN is from a pre-trained ResNet3D-18 model. The CNN component of the CNN-LSTM architecture is from a pre-trained ConvNeXt-Tiny model, while the LSTM component uses a single-layered LSTM with an input size and hidden size of 64. Hyperparameter tuning was done using the validation set as a reference to obtain optimal results that were not overfitting. After hyperparameter tuning, the models were evaluated in the test set using accuracy, F1 score, and inference time as metrics.

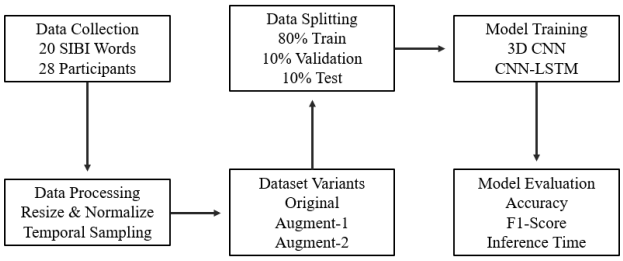

Fig. 1. Methodology Flow.

A. Data Collection

The development of the dataset started with the selection of 20 commonly used words in academic contexts, especially higher education environments. Table I shows the list of words used to build the dataset.

These words were chosen based on their relevance and the availability of gestures in the official SIBI resource. In addition, the words that were chosen contain a mix of nouns, pronouns, question words, and verbs used in the context of formal education. Challenges were encountered during the data collection phase, such as finding a sufficient number of participants and validating that the gestures performed were

correct. Subsequently, 28 participants contributed to the data collection process, where each of them performed the 20 gestures once, resulting in a total of 560 videos in total. This ensures that each gesture has variations in signers and signing style, which can help improve the generalizability of the models. To ensure the quality of the video data, the recordings were ensured to be in landscape orientation and each video had to show the entirety of the subject's body above the torso, as well as hand movements. The recording is also done in different environments, resolution, lighting, and frames per second (FPS) in order to mimic real world conditions as much as possible so as to maintain the model's robustness. Fig. 2 illustrates the diversity of the dataset across these different recording conditions.

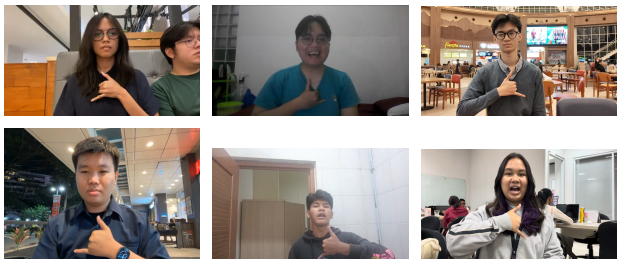

Fig. 2. Examples of dataset diversity across different recording conditions

B. Data Processing

The video data goes through a structured preprocessing pipeline before model training to ensure that the data are ready as inputs to the model. First, each video is decomposed into individual frames using OpenCV, and the default BGR color space is converted to RGB. For the 3D CNN model, all frames are resized to 112×112 pixels, and for CNN-LSTM, frames are resized to 224×224 pixels to maintain uniformity across all videos and speed up training. All frames are normalized based on the pretrained models used. This is done so that the input data distribution matches the expected distribution of the pretrained models used.

1) Temporal Processing: To handle videos of varying lengths to achieve consistent input dimensions for the 3D CNN model, a temporal sampling strategy was implemented. Uniform temporal sampling is used to sample frames from each video sequence such that the resulting video sequence has 32 frames. For videos which are longer than the target sequence length, frames are sampled at evenly distributed intervals across the entire video to preserve the sequential properties of the gesture. For shorter videos, the sequence is padded by repeating the final frame to reach the required length.

As for the CNN-LSTM model, since it can handle varying video lengths as input, the temporal sampling strategy is only used to help speed up the training process at the cost of decreasing the temporal information. The rate at which the temporal sampling is done can be tuned so that the temporal information loss is minor and the model can still converge and generalize to data outside the training set.

2) Data Augmentation: The limitation in vocabulary, dataset size, and real world variability will make it difficult for the result of the model to be generalized. Therefore, data Augmentation is done to artificially increase the size and variety in the dataset to help improve generalization for the model. The data augmentation done can be classified into spatial augmentations and temporal augmentations.

Spatial augmentations are augmentations that affect pixels of each frame, therefore augmenting its spatial features. Spatial augmentations include rotation transformations by +15° and -15°, random horizontal flipping with 0.5 probability, color augmentations that include brightness adjustments (±30%), contrast modifications (±30%), saturation changes (±30%), and hue variations (±10%) to simulate different lighting conditions and camera characteristics, as well as Gaussian blur with kernel size 3 and sigma range from 0.1 to 2.0.

Temporal augmentations are augmentations that affect the number of frames in a video and the way they are sequenced. Slow-motion-like effects are created by duplicating select frames at fixed intervals. Speed-up-like effects are made by sampling every n^{th} frame to simulate faster motion.

To artificially increase the size of the dataset and evaluate the impact of different augmentation strategies, three dataset variants are created before training. The original dataset contains equally resized videos without any augmentation applied, acting as a baseline. Augment-1 variant is a result of augmentation on the original dataset by including two rotated versions of each video (+15° and -15°), tripling the effective dataset size artificially. The Augment-2 dataset expands upon the Augment-1 variation by using two temporal augmentation methods. The first is done by duplicating frames selected at fixed intervals to create a slow-motion effect. The second augmentation is done by sampling every n^{th} frame to simulate faster motion. A summary of the dataset variations are provided in Table II.

TABLE I: Words in the Dataset

Part of Speech	Words
Verb	Tulis
Question Word	Apa, Bagaimana, Berapa, Kapan, Mengapa, Siapa
Noun	Buku, Dosen, Kelas, Kelompok, Kursi, Nilai, Proyek, Tugas
Pronoun	Anda, Dia, Kami, Mereka, Saya

C. Data Splitting and Leakage Prevention

Data splitting is done to prevent information leakage from the training set into the validation and test sets. Because the dataset contains multiple augmented versions of the same base videos, we implemented a data splitting strategy that guarantees that different versions of the same base video do not appear across the different sets after splitting, which would lead to inflated model performance.

TABLE II: Dataset Variants and Applied Processing Step

Dataset Variant	Applied Processing Steps
Original	Resize (112×112 for 3D CNN, 244×244 for CNN-LSTM)
Augment-1	Original + Rotation (+15°, -15°) + Color augmentations
Augment-2	Augment-1 + Slow motion + Frame skipping

Data were split into training, validation, and test sets with an 8:1:1 ratio. The split is stratified by the gestures to ensure that each set maintains equal distribution of gestures. On top of that, the validation and test sets only contain the base videos without their corresponding data-augmented versions, while the training set contains the base videos with their augmented versions. This prevents possible data leakage due to multiple versions of the same video appearing in different sets.

D. Model Architecture

1) 3D CNN Architecture: The 3D CNN model uses a pre-trained ResNet3D-18 (R3D-18) as described in [19], which is a deep residual network designed to be able to handle sequential data, with a total of 18 layers. Table III shows the architecture of ResNet3D-18. The first layer is a convolutional layer with a stride of 1×2×2 to implement a spatial-only downsampling. This is then followed by multiple groups of convolutional residual blocks that steadily increase the number of dimensions while decreasing the size of temporal and spatial dimensions through the use of spatiotemporal downsampling. Residual connections in each block allow the gradients to propagate through the network to help mitigate vanishing gradient. A global spatiotemporal pooling is applied after all the convolutional layers, and the resulting feature vector is then passed to a dense layer with the number of output neurons corresponding to the number of classes. In this study, the dense layer was set to have 20 output units to match the number of gestures present in the SIBI dataset, while all other layers were initialized with weights obtained from pre-training on the Kinetics-400 dataset.

2) Hybrid CNN-LSTM Architecture: The CNN-LSTM architecture is made up of a CNN network to encode spatial features and an LSTM network to encode temporal features. The CNN part of this architecture uses the ConvNeXt-Tiny model, which is a lightweight variant of the ConvNeXt family of models developed by [20]. Table IV shows the architecture of ConvNeXt-Tiny. ConvNeXt is a pure convolutional neural network that builds upon ResNet by complementing it with components inspired by other modern architectures for image recognition, such as the Vision Transformers. The CNN network of this architecture processes every frame of the video sequence individually, resulting in a 768-dimensional vector. This vector is then passed to a dense layer with 64 output units. By the end of this step, a $T \times 64$ vector is produced, where T is the length of the video sequence. The resulting sequence of feature vectors are passed to a single-layered LSTM architecture with 64 hidden units. The hidden state of the last unit of the LSTM is then passed to a dense layer with 20 output units, which corresponds to the number of gestures in the SIBI dataset.

TABLE III: ResNet Architecture Details

Type	**Channels**	**Output Size**	**Kernel**	**Stride**
Conv3d	64	32×56×56	3×7×7	1×2×2
Residual Block x 2	64	32×56×56	3×3×3	1×1×1
Residual Block x 2	128	16×28×28	3×3×3	2×2×2
Residual Block x 2	256	8×14×14	3×3×3	2×2×2
Residual Block x 2	512	4×7×7	3×3×3	2×2×2
Global Average Pooling	512	1×1×1	-	-
Linear	-	20	-	-

TABLE IV: ConvNeXt-Tiny & LSTM Architecture Details

Type	**Channels**	**Output Size**	**Kernel**	**Stride**
Conv2d	96	T×56×56	4×4	4
ConvNeXt Block ×3	96	T×56×56	7×7	1
Conv2d	192	T×28×28	2×2	2
ConvNeXt Block ×3	192	T×28×28	7×7	1
Conv2d	384	T×14×14	2×2	2
ConvNeXt Block ×9	384	T×14×14	7×7	1
Conv2d	768	T×7×7	2×2	2
ConvNeXt Block ×3	768	T×7×7	7×7	1
Global Avg Pooling	768	T×1×1	-	-
Temporal Modelling				
Linear	-	T×64	-	-
LSTM	-	64	-	-
Linear	-	20	-	-

E. Model Evaluation

Both models were evaluated on two aspects: their performance on the classification of data on the test set and their computational efficiency. The former will be measured by using the F1 score, and also accuracy. The latter will be quantified by the average inference time of each model per frame per millisecond (ms). To ensure measurement accuracy, the models will be running on warm-up iterations before doing the main inference.

IV. Results and Discussion

Table V shows the accuracy and F1 scores on the test set for both model architectures across each dataset variant. Table VI

shows the average inference time for each model architecture evaluated on the test data.

TABLE V: Model Performance Across Dataset Variations

Model	Dataset Variation	Accuracy	F1 Score
3D CNN	Original	0.857	0.855
	Augment-1	0.928	0.917
	Augment-2	0.875	0.854
CNN-LSTM	Original	0.982	0.982
	Augment-1	0.982	0.973
	Augment-2	0.982	0.980

TABLE VI: Average Inference Time per Frame

Model	Time (ms)
3D CNN	0.74
CNN-LSTM	2.70

A. 3D CNN Performance Analysis

The 3D CNN model obtained the best results on the Augment-1 dataset variant with an accuracy of 0.928 and an F1 score of 0.917. This was an improvement compared to its performance on the original dataset, where it had an accuracy of 0.857 and an F1 score of 0.855. This suggest that geometric augmentations are effective in artificially increasing the size of the dataset without hindering the model's ability to generalize. In contrast, the model's performance decreased on the Augment-2 variant, achieving an accuracy of 0.875 and an F1 score of 0.854. This may show that temporal augmentations may disrupt the model's ability to capture relevant features from the sequential data.

Table VII shows the F1 scores for every word in the test set for all dataset variations. Models trained on the original dataset and the Augment-2 variant struggled with the word "Dia", as they obtained an F1 score of 0.40 and 0.00, respectively. Meanwhile, the model trained on the Augment-1 achieved an F1 score of 1.00 on the word "Dia".

B. CNN-LSTM Performance Analysis

The CNN-LSTM model achieved similar results across all dataset variants. It consistently achieved accuracies and F1 scores above 0.97 with only small differences, regardless of whether spatial or temporal augmentations were applied. This consistency shows that the combination of CNN as a frame encoder to extract spatial information and LSTM as a temporal information extractor allows the model to handle both geometric and temporal variations effectively, making it highly robust and adaptable.

Table VIII shows the F1 scores for every word in the test data for all dataset variations. The CNN-LSTM models achieved perfect F1 scores on most of the words in the test data for every dataset variant. Moreover, the words with non-perfect F1 scores differ between every dataset variant, indicating that the CNN-LSTM model has no difficulty predicting a particular word.

TABLE VII: F1 Score for every Word of all 3D CNN Models

Word	Dataset Variant		
	Original	Augment-1	Augment-2
Anda	1.00	0.80	1.00
Apa	1.00	0.67	0.80
Bagaimana	0.80	1.00	0.86
Berapa	0.80	1.00	0.80
Buku	1.00	0.86	0.86
Dia	0.40	1.00	0.00
Dosen	0.75	1.00	0.80
Kami	1.00	1.00	0.86
Kapan	0.80	1.00	1.00
Kelas	0.86	1.00	1.00
Kelompok	0.80	1.00	0.67
Kursi	1.00	0.80	1.00
Mengapa	0.86	1.00	1.00
Mereka	0.75	0.75	0.86
Nilai	1.00	0.80	0.80
Proyek	1.00	0.67	0.80
Saya	0.80	1.00	1.00
Siapa	0.50	1.00	1.00
Tugas	1.00	1.00	1.00
Tulis	1.00	1.00	1.00

TABLE VIII: F1 Score for every Word of all CNN-LSTM Models

Word	Dataset Variant		
	Original	Augment-1	Augment-2
Anda	1.00	1.00	1.00
Apa	1.00	0.80	1.00
Bagaimana	1.00	1.00	1.00
Berapa	1.00	1.00	1.00
Buku	1.00	1.00	1.00
Dia	1.00	1.00	1.00
Dosen	0.86	1.00	1.00
Kami	1.00	1.00	1.00
Kapan	1.00	1.00	1.00
Kelas	1.00	1.00	1.00
Kelompok	0.80	1.00	1.00
Kursi	1.00	1.00	1.00
Mengapa	1.00	1.00	0.80
Mereka	1.00	1.00	1.00
Nilai	1.00	1.00	1.00
Proyek	1.00	0.67	1.00
Saya	1.00	1.00	1.00
Siapa	1.00	1.00	0.80
Tugas	1.00	1.00	1.00
Tulis	1.00	1.00	1.00

C. Comparative Analysis

Comparing the two approaches, the CNN-LSTM outperformed the 3D CNN across all dataset variants. The difference in scores achieved ranged from 0.054 to 0.125 in accuracy and from 0.056 to 0.128 in F1 score. The smallest gap appeared on the Augment-1 dataset, where both models benefited from geometric augmentation, but the CNN-LSTM still obtained better scores. The largest gaps were between the original and Augment-2 datasets, which shows the CNN-LSTM's robustness to both unaltered and temporally augmented data.

Although the CNN-LSTM achieved better scores compared

to the 3D CNN overall, the 3D CNN model obtained lower inference times when compared to the CNN-LSTM model. The 3D CNN processed video frames approximately 3.6 times faster than the CNN-LSTM, with average inference times of 0.74 milliseconds per frame compared to 2.70 milliseconds. The measured inference times represent isolated frame processing under optimized conditions and serve as indicators of the models' computational complexity. However, these metrics do not directly translate to real-time system performance, as discussed in the following section.

D. Real-Time Performance Analysis

To evaluate the practicality of the CNN-LSTM model for real-time applications, an real-time inference performance test was done. The model was ran on three laptop configurations with different computational power. Table IX presents the hardware specifications used for real-time inference performance test.

TABLE IX: Hardware Specifications Across Laptop Configurations

Device	**Processor**	**GPU**	**VRAM**
Laptop 1	Ryzen 7 6800HS	RTX 3050Ti	4GB
Laptop 2	i5-13500HX	RTX 4050	6GB
Laptop 3	Ultra 9-275HX	RTX 5090	16GB

The performance evaluation across these configurations revealed insights into the computational requirements for real-time gesture recognition. As shown in Fig. 3, there is a clear linear relationship between hardware capability and inference performance, with GPU acceleration providing a consistent 2.4-2.5× speedup across all devices. CPU-only execution yielded frame rates between 4-7 FPS across all tested configurations, substantially below the 24-30 FPS threshold typically required for smooth, real-time video processing applications. Even with GPU acceleration, the most powerful configuration (Laptop 3 with RTX 5090) achieved only 17 FPS, falling short of real-time requirements.

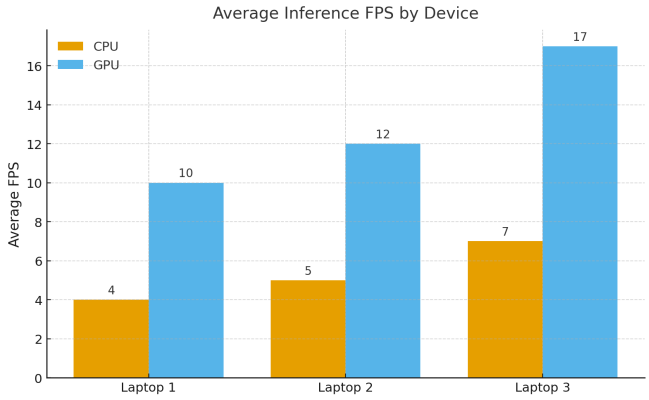

Fig. 3. FPS trends across devices for CPU and GPU inference.

Despite these performance constraints, the system demonstrates practical viability for deployment scenarios where slight delays are acceptable. Fig. 4 illustrates the deployed real-time inference application, showing that the system processed live video input and output gesture predictions with performance metrics displayed in real-time. This implementation validates the model's functional deployment while highlighting the performance trade-offs discussed above.

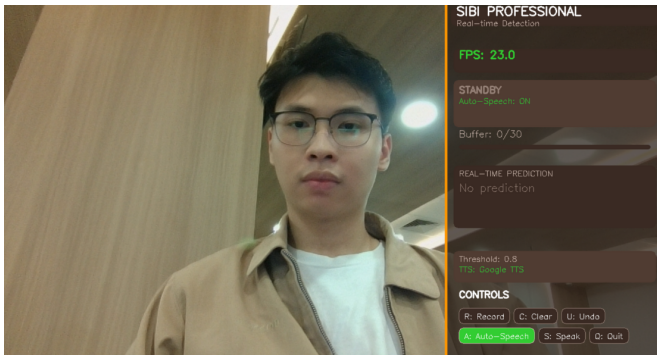

Fig. 4. Real-time SIBI gesture recognition application interface.

However, a problem arised during real-time testing where the model would occasionally make predictions when no gesture, which leads to a false positive. This is evident for the words "Apa", "Mengapa", and "Dia" because of their similar starting positions. This problem can be mitigated by increasing the confidence threshold for prediction outputs to ensure that the model has high enough confidence before making predictions

For real-world applications, model compression techniques can be used to improve inference speed without sacrificing accuracy significantly. One way this can be done is through quantization, which is technique that changes the model's weight representations with lower precision data types such as FP16 or INT8, hence reducing the computational and memory costs of the model. For situations where speed of inference is prioritized and minor decreases in accuracy is acceptable, the 3D CNN model can be a viable option due to its faster inference speed. However, this model comes with the downside of having a fixed-length input requirement which limits its flexibility in handling sequences with varying lengths.

V. Conclusion

Two neural network architectures for SIBI sign language recognition were studied. One of them was a pre-trained ResNet3D-18 and the other was a hybrid CNN-LSTM model using a pre-trained ConvNeXt-Tiny as the CNN backbone. The CNN-LSTM model consistently achieved better scores compared to the 3D CNN across all dataset variations. Its high accuracy and F1 scores show the capability of the CNN-LSTM model for sign language recognition, as well as its robustness to data augmentation. As such, the CNN-LSTM model is a suitable option for sign recognition tasks where accuracy is the main concern. However, the 3D CNN's faster inference speed

makes it an alternative option for deployment in environments where faster processing time is more important and small loss in accuracy is acceptable.

Future research should focus on exploring lightweight variants of the CNN-LSTM architecture to obtain faster inference time. Model compression techniques can be explored to reduce hardware requirements on real-world applications. Future research could also look into transformer-based models as an alternative to LSTMs that could potentially offer an increase in both speed and performance. The number of gestures as well as sample sizes could also be increased to help improve model generalization and increase real-world application viability. On top of that, additional input modalities could be integrated such as depth information.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Bina Nusantara University and to Lili Ayu Wulandhari for providing the resources, funding, guidance, and supervision that made this research possible.

AUTHOR CONTRIBUTIONS

Marvel Martawidjaja and Richie Darylson Tandiono jointly conducted the entire research process, including dataset design, data preprocessing, model development, experimentation, and manuscript preparation. Lili Ayu Wulandhari served as the academic supervisor and provided guidance, methodological advice, and critical review of the manuscript.

DATA AVAILABILITY

The participants of this study did not provide written consent for their data to be shared publicly. Due to the sensitive nature of the dataset involving sign language videos, the supporting data are not available.

REFERENCES

[1] Indonesian Ministry of Education, "Kamus Sistem Isyarat Bahasa Indonesia (SIBI)," pmpk.kemdikbud.go.id. https://pmpk.kemdikbud.go.id/sibi/ (accessed Jul. 10, 2025).

[2] R. A. Mursita, "Respon tunarungu terhadap penggunaan Sistem Bahasa Isyarat Indonesia (SIBI) dan Bahasa Isyarat Indonesia (BISINDO) dalam komunikasi," *INKLUSI*, vol. 2, no. 2, pp. 221–240, Dec. 2015, doi: 10.14421/ijds.2202.

[3] Indonesian Center for Women in Arts, "Indonesia: One language, many voices," ICWA. https://www.icwa.org/indonesia-one-language/ (accessed Jul. 10, 2025).

[4] I. Papastratis, C. Chatzikonstantinou, D. Konstantinidis, K. Dimitropoulos, and P. Daras, "Artificial intelligence technologies for sign language," *Sensors*, vol. 21, no. 17, p. 5843, Aug. 2021, doi: 10.3390/s21175843.

[5] F. A. Setiawan, Z. A. Rohmah, and G. F. Laxmi, "Indonesian sign language (BISINDO) alphabet detection using the You Only Look Once (YOLO) algorithm version 8," in *Proc. Int. Conf. Comput., Control, Inform. Appl. (IC3INA)*, Oct. 2024, pp. 388–393, doi: 10.1109/IC3INA64086.2024.10732209.

[6] R. Setiawan, Y. Yunita, F. F. Rahman, and H. Fahmi, "BISINDO (Bahasa Isyarat Indonesia) sign language recognition using deep learning," *IT for Society*, vol. 9, no. 1, Mar. 2024, doi: 10.33021/itfs.v9i1.5076.

[7] M. Z. Fauzi, R. Sarno, and S. C. Hidayati, "Recognition of real-time BISINDO sign language-to-speech using machine learning methods," in *Proc. Int. Conf. Comput. Sci., Inf. Technol. Eng. (ICCoSITE)*, Feb. 2023, pp. 986–991, doi: 10.1109/ICCoSITE57641.2023.10127743.

[8] M. A. Limantara and D. Tristianto, "SIBI alphabet detection system based on convolutional neural network (CNN) method as learning media," *Internet of Things and Artificial Intelligence Journal*, vol. 4, no. 1, pp. 143–161, Mar. 2024, doi: 10.31763/iota.v4i1.716.

[9] R. A. T. Kurniawan and W. Kaswidjanti, "SIBI dynamic gesture translation using MediaPipe and long short-term memory in real-time," in *Proc. 1st Int. Conf. Adv. Inform. Intell. Inf. Syst. (ICAI3S)*, Feb. 2024, pp. 49–60, doi: 10.2991/978-94-6463-366-5_6.

[10] J. Huang and V. Chouvatut, "Video-based sign language recognition via ResNet and LSTM network," *J. Imaging*, vol. 10, no. 6, p. 149, Jun. 2024, doi: 10.3390/jimaging10060149.

[11] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden, "Sign language transformers: Joint end-to-end sign language recognition and translation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10023–10033, doi: 10.1109/CVPR42600.2020.01004.

[12] R. Cui, H. Liu, and C. Zhang, "A deep neural framework for continuous sign language recognition by iterative training," *IEEE Trans. Multimedia*, vol. 21, no. 7, pp. 1880–1891, Jul. 2019, doi: 10.1109/TMM.2018.2886712.

[13] N. Kolotouros, G. Pavlakos, and K. Daniilidis, "Convolutional mesh regression for single-image human shape reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, May 2019, pp. 4501–4510, doi: 10.1109/CVPR.2019.00463.

[14] A. N. Handayani et al., "Hand keypoint-based CNN for SIBI sign language recognition," *Int. J. Robot. Control Syst.*, vol. 5, no. 2, pp. 813–824, Apr. 2025, doi: 10.31763/ijrcs.v5i2.1745.

[15] 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, Sep. 2021, doi: 10.1109/ACCESS.2021.3110912.

[16] D. K. Singh, "3D-CNN based dynamic gesture recognition for Indian sign language modeling," *Procedia Comput. Sci.*, vol. 189, pp. 76–83, 2021, doi: 10.1016/j.procs.2021.05.071.

[17] F. Anvarov, D. H. Kim, and B. C. Song, "Action recognition using deep 3D CNNs with sequential feature aggregation and attention," *Electronics*, vol. 9, no. 1, p. 147, Jan. 2020, doi: 10.3390/electronics9010147.

[18] J. Chen, J. Wang, Q. Yuan, and Z. Yang, "CNN-LSTM Model for Recognizing Video-Recorded Actions Performed in a Traditional Chinese Exercise," *IEEE J. Transl. Eng. Health Med.*, vol. 11, pp. 351–359, Jun. 2023, doi:10.1109/JTEHM.2023.3282245.

[19] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A close look at spatiotemporal convolutions for action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6450–6459, doi: 10.1109/CVPR.2018.00675.

[20] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11976–11986, doi: 10.1109/CVPR52688.2022.01167.