

Optimization of Convolutional Neural Network Algorithm for Indonesian Sign Language Classification

Alvin Bintang Rebrastya¹, Sumarni Adi^{2*}, Hanif Al Fatta³,
Windha Mega Pradnya Dhuhi⁴, Ika Nur Fajri⁵, Muhammad Hanafi⁶

Department of Informatics, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia¹
Faculty of Computer Science, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia^{2, 3, 4, 5}
Magister of Informatics Engineering, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia⁶

Abstract—Sign language serves as a primary mode of communication for individuals who are deaf or speech impaired, using hand gestures to convey meaning visually. While it facilitates communication among the deaf community, it presents challenges for interaction with those who rely on spoken language. This study aims to recognize hand signs representing the letters A to Y (excluding J and Z) in the Indonesian Sign Language (SIBI) using image-based input. A custom dataset was collected through personal photo shoots and used to train a Convolutional Neural Network (CNN) implemented in Python using the TensorFlow library. The study also focuses on optimizing the CNN architecture to achieve high classification accuracy. Evaluation using a confusion matrix on the test data resulted in an overall accuracy of 87.1%, while real-time testing achieved an accuracy of 90.25%. The number of convolutional filters and dropout rates was adjusted to prevent underfitting and overfitting during model training.

Keywords—Indonesian Sign Language; hand sign recognition; image classification; Convolutional Neural Network

I. INTRODUCTION

According to the World Health Organization, more than 5% of the global population, approximately 430 million people, experience hearing loss greater than 35 decibels (dB) [1]. In Indonesia, around 7.03% of individuals with disabilities are categorized as deaf [2]. Deaf individuals use sign language to communicate, forming words and sentences through hand gestures. Sign language is a visual-gestural language that relies on movement and perception through vision [3]. It enables communication within the deaf community; however, interactions with people who rely on spoken language remain challenging.

In Indonesia, two main sign language models are used: the Indonesian Sign System (SIBI) and the Indonesian Sign Language (BISINDO). SIBI employs right-hand movements to represent alphabetical characters and is officially standardized by the Indonesian Ministry of Education and Culture [4]. BISINDO, in contrast, uses both hands and has developed organically within Indonesian society, resulting in regional variations. While BISINDO is widely used informally, SIBI remains the formal and standardized model.

Recent advancements in deep learning technology offer potential solutions for automating sign language recognition. One prominent algorithm in this domain is the Convolutional Neural Network (CNN), a type of deep learning model designed to process visual data [5]. Despite many studies applying CNNs for sign language recognition in various countries [6–11], there is limited research focusing on Indonesian Sign Language, particularly the SIBI model, using custom datasets collected in varied real-world conditions. This gap highlights the need for approaches tailored to the Indonesian context.

This study aims to address this gap by developing and optimizing a CNN-based model for classifying 24 SIBI hand signs (excluding J and Z). The model is trained on a custom image dataset collected via smartphone and webcam. The objectives of this research are to:

- 1) Design a CNN architecture suitable for classifying Indonesian SIBI signs.
- 2) Evaluate the impact of convolutional filter configurations and dropout rates on model performance.
- 3) Validate the model's accuracy through offline and real-time testing using confusion matrix analysis.

By providing a lightweight, efficient, and accurate CNN model trained on a custom dataset, this study contributes to improving accessibility and communication support tools for the Indonesian deaf community.

II. RELATED WORKS

Various kinds of previous studies have been proposed to recognize sign language. Sign languages in various countries have different signs, such as studies using Indian [6], Pakistani [7], Arabic [8], Bangla [9], Brazilian (Libras) [10], and Indonesian Sign Language with the BISINDO model [11].

In [6], A CNN-based Indian Sign Language (ISL) recognition system was developed to enable real-time hand gesture detection via a camera, achieving over 90% accuracy and was deployed on mobile devices using TensorFlow Lite and Flutter to enhance communication accessibility for the deaf-mute community. Research [7] Principal Component Analysis (PCA) and K-Nearest Neighbors (KNN) were applied, achieving 85% accuracy on test data and 80% when evaluated via webcam. Research [8] utilized the Faster R-CNN method.

*Corresponding author.

Using the ResNet-18 architecture yielded an accuracy of 93.4%, while VGG-16 achieved 93.2%. The work in [9] employed the YOLOv4 object detection model, obtaining an overall detection accuracy of 97.95%. In [10], a CNN-based approach with the VGGNet architecture was implemented. In [17], the study explored several input image sizes: 32×32 pixels (97.98% accuracy), 64×64 (99.42%), 128×128 (97.40%), and 256×256 (86.99%). Study [11] also adopted a CNN method, comparing multiple architectures. The modified version of AlexNet achieved an accuracy of 98.6%, while the modified VGG-16 architecture underperformed with only 3.8%. A novel architecture introduced in the same study, referred to as the C model, reached 98.3% accuracy.

Although various deep learning models have been applied in previous studies, such as Faster R-CNN [8], YOLOv4 [9], and modified architectures of AlexNet and VGG [10] [11], most of them require complex setups or large training resources. Furthermore, these models were developed for different language systems and may not generalize well to the Indonesian context. Compared to these methods, CNN offers a balanced trade-off between accuracy, computational efficiency, and ease of implementation.

This study focuses on a lightweight CNN model that is suitable for small-scale deployment with limited GPU resources. Our approach eliminates the need for extensive pre-training or large-scale datasets, making it more feasible for real-world applications in Indonesia.

III. METHODS

The research methodology in this study consists of several stages: data collection, data transformation, data splitting, CNN model training, prediction, and evaluation. An overview of these stages is illustrated in Fig. 1.

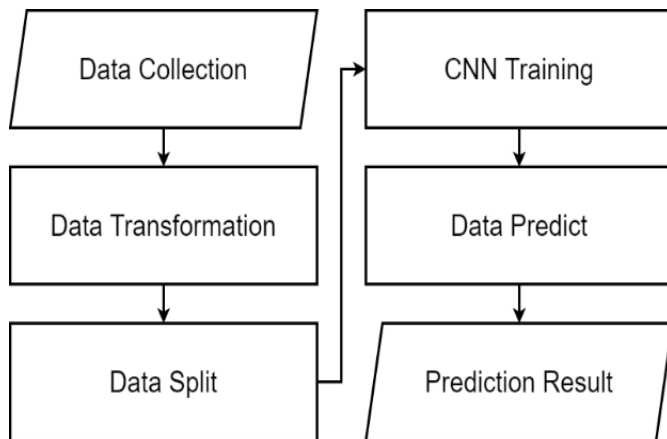


Fig. 1. Research stages.

The study began with the collection of sign language data. A total of 5,280 images were captured through self-recording using smartphones and webcam cameras. The next stage involved data transformation, which included resizing, grayscale conversion, data augmentation, and normalization.

Once transformed, the data were divided into two main sets: training and testing data. The core phase of the study was the training of a CNN model using the prepared dataset. Following

training, the model was evaluated to assess its accuracy. Subsequently, testing was conducted using both the test dataset and newly captured images from a webcam to evaluate the model's ability to classify and predict sign language gestures in real time.

The following are some basic points that will be used in this research:

A. Data

The data used in this study consists of images representing Indonesian Sign Language (SIBI) hand gestures using the right hand. The dataset was collected independently [12], capturing variations in hand position, lighting, and background. The devices used for data collection included smartphone cameras and laptop webcams. The full dataset is accessible online [12].

B. Convolutional Neural Network

Convolutional Neural Networks (CNNs) are widely used for tasks such as speech recognition, image classification, and 3D object recognition. A typical CNN architecture includes two main components: a feature learning component and a fully connected layer. The feature learning section consists of convolutional layers, ReLU activation functions, and pooling layers. The fully connected layer is composed of neurons that process the extracted features to perform final classification. An overview of the general CNN architecture is presented in Fig. 2.

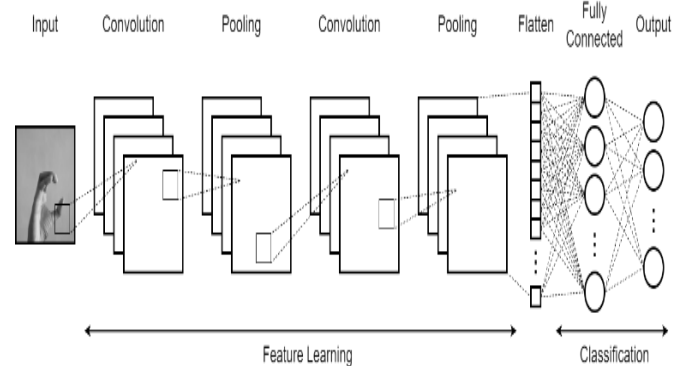


Fig. 2. Architecture of the CNN.

Convolution is the fundamental operation in CNNs, involving a matrix-based filtering process. This operation applies a filter (or kernel) matrix to an input image matrix to extract features, as defined in [13]:

$$\text{net}[i, j] = (x * w)[i, j] = \sum_m \sum_n x[m, n] w[i - m, j - n] \quad (1)$$

In this equation, w denotes the image matrix and x denotes the filter matrix. The convolution of these matrices results in a new feature map representing the output image. Indices i and j refer to the image matrix, while m and n refer to the kernel.

This study proposes a simple architecture consisting of the following layers:

- Convolutional layer 0,
- Max pooling layer 0,
- Convolutional layer 1,

- Dropout layer 0,
- Convolutional layer 2,
- Max pooling layer 1,
- Dropout layer 1,
- Flatten layer
- Two fully connected layers.

This proposed architecture is illustrated in Fig. 3.

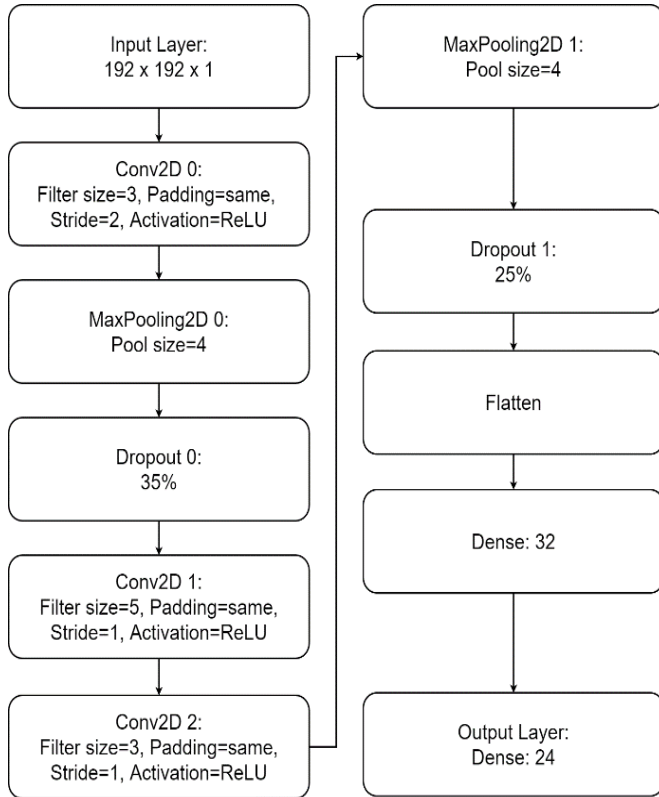


Fig. 3. Architecture model.

1) *Baseline comparison and validation design:* To validate the effectiveness of the proposed CNN architecture, we compared it with a basic CNN baseline model using fewer layers and no dropout. Both models were trained on the same dataset using identical preprocessing and training parameters. The baseline model consisted of a single convolutional layer followed by pooling and a fully connected layer.

Performance comparison between the proposed and baseline models is summarized in Table II. The proposed model achieved a higher accuracy (87.1%) compared to the baseline model (74.5%), and showed better generalization, as indicated by closer training and validation accuracy curves. Furthermore, F1-score, precision, and recall metrics improved by more than 10% on average per class.

This result confirms that the architectural optimizations—particularly in the number of convolutional filters and dropout rates—significantly improved classification performance and minimized overfitting.

C. Evaluation

The proposed CNN model was evaluated using a confusion matrix. This matrix produces four types of values: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These values were used to calculate performance metrics, including accuracy, precision, recall, and F1-score for each class. The confusion matrix serves as a basis for assessing the classification model's performance. Table I presents the layout of the confusion matrix [14].

TABLE I. CONFUSION MATRIX

Prediction	Actual Value	
	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (TN)	True Negative (TN)

Definitions:

- True Positive (TP): Positive instances correctly classified.
- True Negative (TN): Negative instances correctly classified.
- False Positive (FP): Negative instances incorrectly classified as positive.
- False Negative (FN): Positive instances incorrectly classified as negative.

Based on these values, the following metrics were computed:

1) Overall Accuracy: Proportion of correct predictions out of total predictions [15][16]:

$$\text{Overall Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total Number of predictions}} \quad (2)$$

2) Per-class Accuracy: Accuracy calculated for each class individually (macro average), while overall accuracy is a micro average [15].

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

3) Precision: Measures the accuracy of positive predictions.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

4) Recall (Sensitivity): Measures the model's ability to correctly detect positive instances.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (5)$$

5) F1-score: The harmonic mean of precision and recall.

$$F1 - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (6)$$

IV. RESULTS

A. Dataset Collection

The dataset used in this study is privately collected, comprising 5,280 images captured through self-recording [12].

The data consist of two types of images: 1) high-resolution RGB images (3000×3000 pixels) taken using a smartphone camera, and 2) lower-resolution grayscale images (250×250 pixels) taken using a webcam.

The dataset includes 24 hand gestures representing the SIBI alphabet (letters A to Y, excluding J and Z), with 220 images per class. An example of the dataset is shown in Fig. 4.

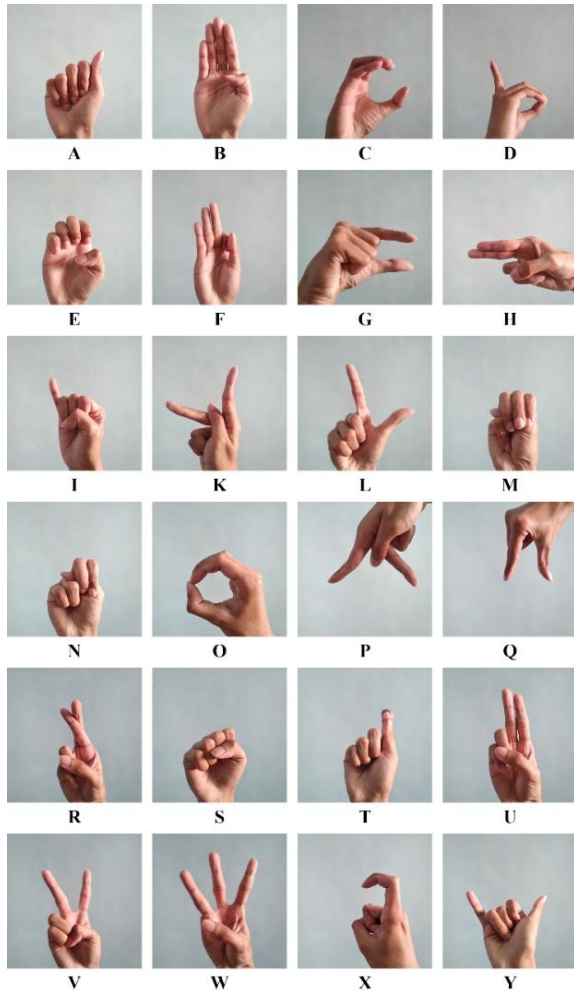


Fig. 4. Data sample.

B. Data Transformation

This stage involves transforming the data into a format suitable for training. The following techniques were applied:

1) *Resize*: All images were resized to 192×192 pixels and converted to grayscale to reduce computational load during training.

2) *Grayscale*: RGB images, composed of Red, Green, and Blue layers, were converted to grayscale using a weighted average of RGB channels. The grayscale transformation process is illustrated in Fig. 5 and Fig. 6.

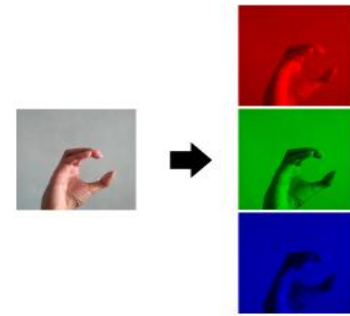


Fig. 5. RGB Data sample

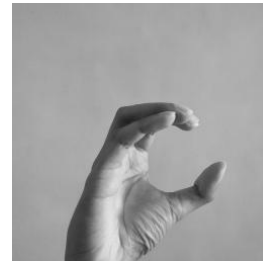


Fig. 6. Grayscale image.

3) *Data augmentation*: Augmentation techniques were applied to increase dataset diversity. Thirty images per class were selected for transformation using random rotation (0.085), random zoom (0.1), and random translation (0.15). Images exceeding frame bounds were padded with black (zero constant). Sample augmentation results are shown in Fig. 7.

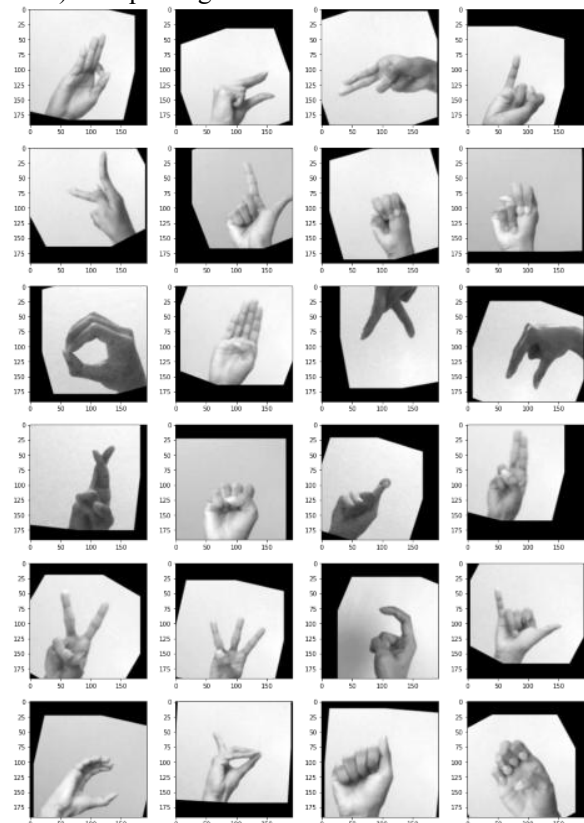


Fig. 7. Augmentation data sample.

C. Data Split

After augmentation, the dataset increased to 6,000 images (5,280 original + 720 augmented). The data were split as follows: 68% for training, 17% for validation, and 15% for testing.

D. Training Process

The CNN model was trained using a computer with an Intel Core i5-10210U CPU and NVIDIA GeForce MX330 GPU (2GB VRAM, GDDR5). A total of 720 images were used for testing.

Five different CNN configurations were evaluated by adjusting the number of convolution filters and dropout rates, while keeping filter size, stride, and padding constant. Table II lists the model architectures tested. The training used several parameters shown in Table III.

TABLE II. TRAINING PARAMETERS

Parameter	Value
Optimizer	Adam
Learning Rate	0.001
Loss Function	Categorical Cross-entropy
Epochs	80

TABLE III. MODEL ARCHITECTURE TESTING

No.	Parameter				
	Conv0	Dropout0	Conv1	Conv2	Dropout1
1	48	-	96	192	-
2	24	-	48	96	-
3	24	0,35	48	96	0,25
4	48	0,35	96	192	0,25
5	24	0,85	48	48	0,75

E. Testing Model

Testing the model from the CNN algorithm that has been made, then testing using a confusion matrix by testing the image from the test data, as many as 15% or 900 data units. This test was carried out five times to determine the level of accuracy, duration of training, accuracy graph, loss graph, and whether the two graphs were overfit or underfit. The parameters that are changed are the number of convolution filters and dropouts, while the filter size, stride, and padding in the filter are not changed. Table IV is the result of a comparison of the training that has been done.

F. Real-Time Testing

The architecture used for real-time testing corresponds to configuration number 3 in Table IV, which previously demonstrated the best performance. The testing was conducted using the right hand to perform SIBI (Sistem Isyarat Bahasa Indonesia) sign language gestures within the detection area of a standard laptop webcam. The system was implemented as a desktop application.

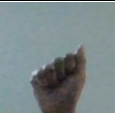



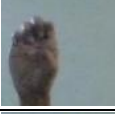





The testing results, including accuracy per class, are listed in Table V. The model achieved an average accuracy of 90.25%.















However, lower accuracy was observed for the letters M and N, which can be attributed to their visual similarity, making them more difficult to distinguish during classification.

TABLE IV. TRAINING COMPARISON RESULTS

No.	Accuracy	Training Duration
1	82,4	10m 39,3s
2	80,0	6m 8,4s
3	87,1	6m 20,4s
4	90,3	11m 6,2s
5	46,6	5m 44,9s

TABLE V. REAL-TIME TEST RESULTS

Input	Class	Result (%)
	A	89
	B	97
	C	98
	D	96
	E	91
	F	89
	G	96
	H	100
	I	88
	K	97

Input	Class	Result (%)
	L	96
	M	70
	N	63
	O	93
	P	84
	Q	84
	R	89
	S	91
	T	90
	U	94
	V	91
	W	93
	X	96
	Y	91

V. DISCUSSION

The results of the training, along with the corresponding graphs and confusion matrix shown in Table III, are presented as follows:

1) *First test:* In this test, the model configuration corresponds to Test 1 in Table IV. The training and validation accuracy graphs, along with the confusion matrix, are shown in Fig. 8 and Fig. 9. The graph indicates signs of overfitting, as the validation accuracy plateaus after the 20th epoch and exhibits a substantial gap compared to the training accuracy. Additionally, the loss graph shows a noticeable increase starting from the 45th epoch, further confirming the presence of overfitting.

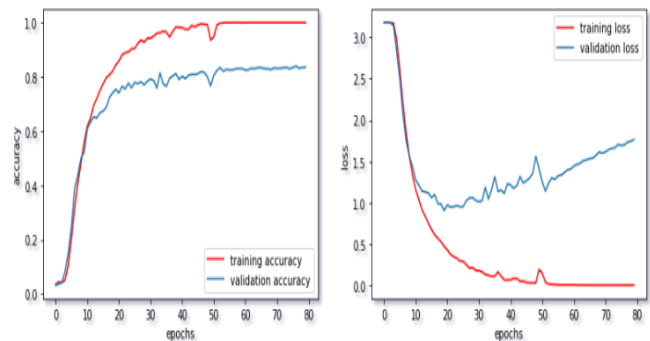


Fig. 8. Graph of the first test result.

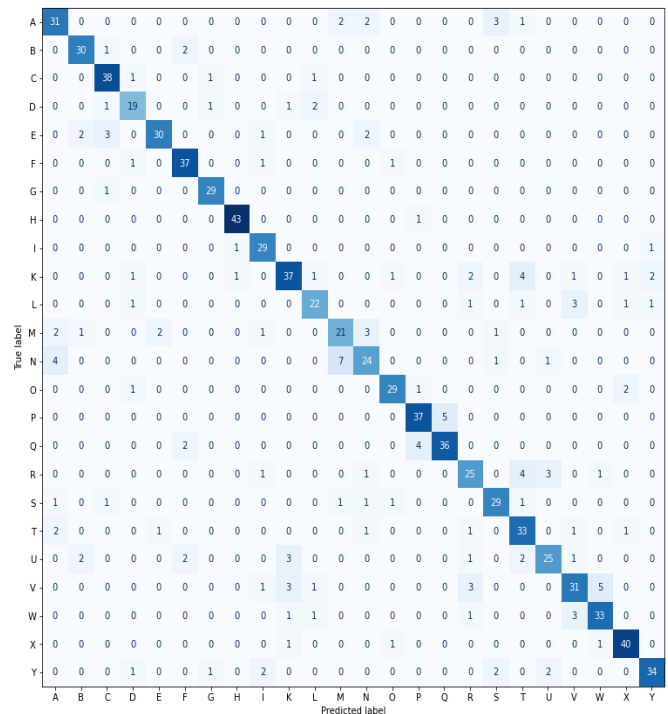


Fig. 9. Confusion matrix for the first test.

2) *Second test:* This test uses the configuration listed as Test 2 in Table IV. The training and validation graph, as well as the confusion matrix are presented in Fig. 10 and Fig. 11. According to [18], overfitting can occur when the neural

network fails to improve its performance during training. In this case, signs of overfitting appear after the 20th epoch, where validation accuracy stagnates and deviates significantly from training accuracy. The loss curve also rises steadily without decreasing as training progresses.

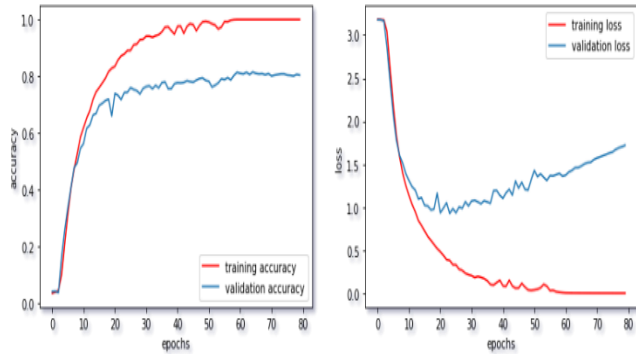


Fig. 10. Graph of the second test result.

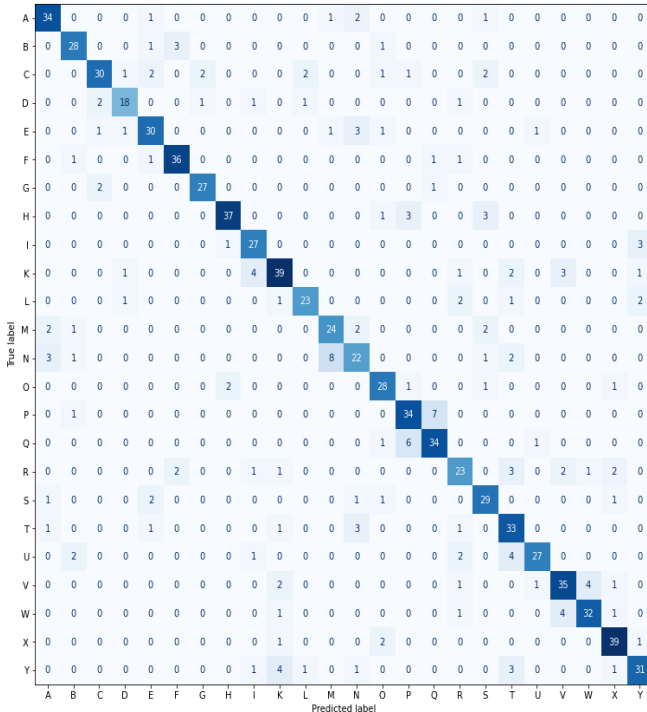


Fig. 11. Confusion matrix for the second test.

3) *Third test:* The configuration for this test is listed as Test 3 in Table IV. The training and validation graphs, along with the confusion matrix, are shown in Fig. 12 and Fig. 13. In this test, validation accuracy closely follows training accuracy, and the validation loss also aligns well with training loss throughout the training process. This indicates that the model avoids both underfitting and overfitting, resulting in high and stable accuracy.

4) *Fourth test:* The model configuration in this test is listed as Test 4 in Table IV. The corresponding graphs and confusion matrix are presented in Fig. 14 and Fig. 15. As noted in [9], convolutional layers with a large number of filters tend to require longer training times. This configuration yields the highest accuracy among the tests; however, slight overfitting is observed after the 25th epoch, where the validation accuracy shows minimal improvement.

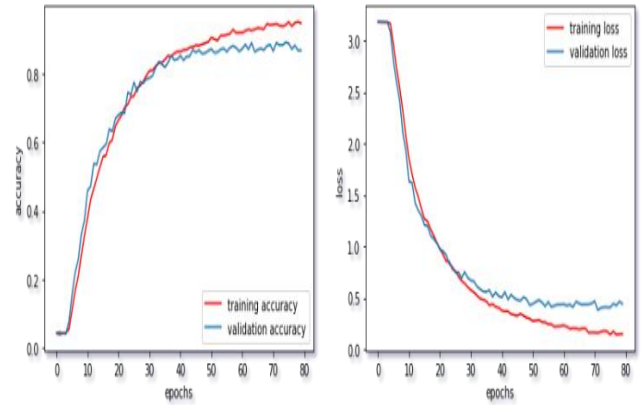


Fig. 12. Graph of the third test result.

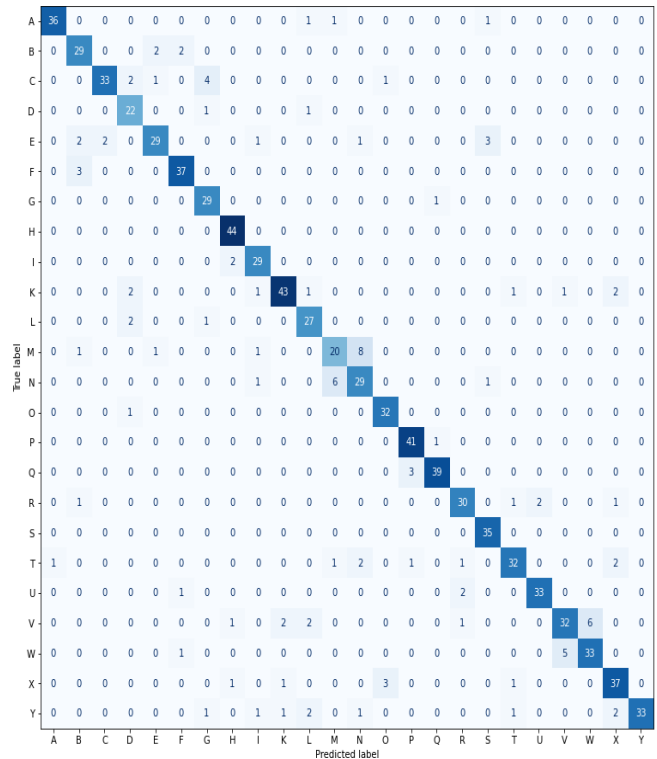


Fig. 13. Confusion matrix for the third test.

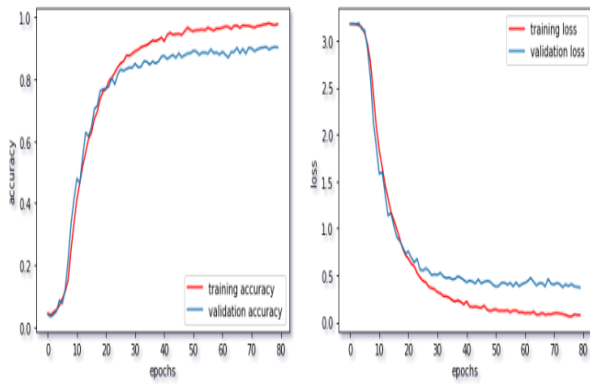


Fig. 14. Graph of the fourth test result.

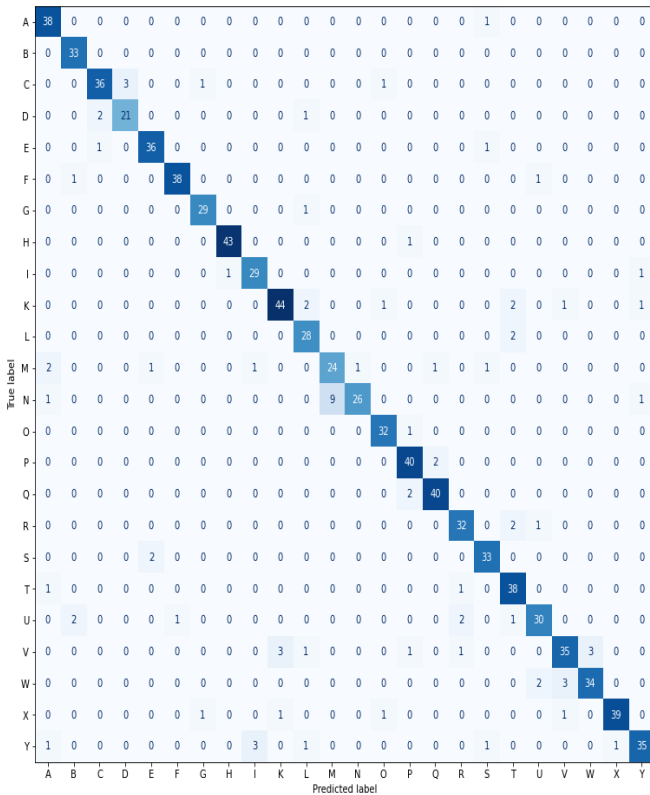


Fig. 15. Confusion matrix for the fourth test.

5) Fifth Test: This test uses the configuration listed as Test 5 in Table IV. The corresponding training graphs and confusion matrix are shown in Fig. 16 and Fig. 17. This model employs the lowest number of filters among all configurations. As stated in [13], network performance can be influenced by the depth of the architecture. In this test, the model achieves the lowest accuracy and the highest loss. The graph indicates underfitting, as the validation accuracy exceeds the training accuracy, suggesting the model is not learning effectively.

Among the five model architectures tested in Table IV, Test 3 yields the best performance, achieving high accuracy without signs of overfitting or underfitting. The final confusion matrix used to calculate per-class accuracy, precision, recall, and F1-score is presented in Fig. 18. While this figure demonstrates

high overall accuracy, the precision, recall, and F1-score values are relatively lower, indicating potential room for improvement in class-level predictions.

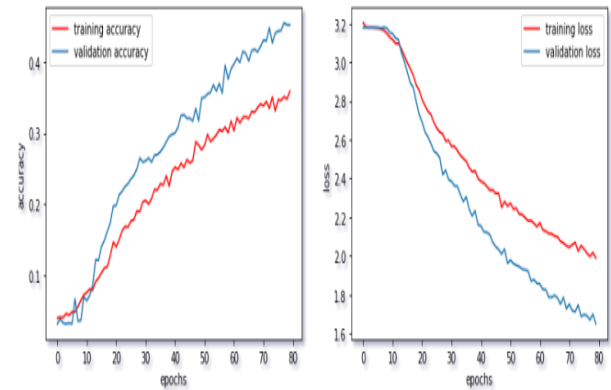


Fig. 16. Graph of the fifth test result.

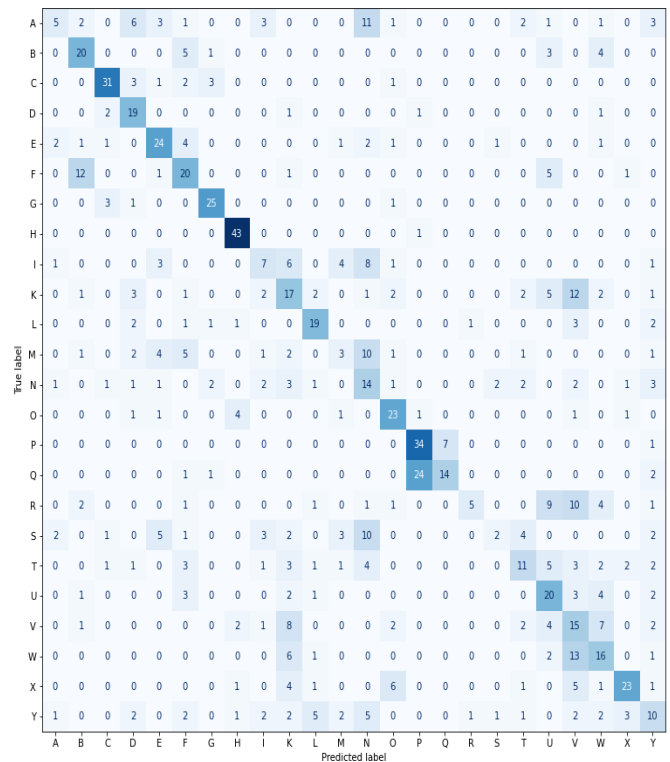


Fig. 17. Confusion matrix for the fifth test.

Compared to the baseline CNN model without architectural optimization, the proposed model demonstrates a significant improvement in both accuracy and robustness. While the baseline model produced acceptable results for simple gesture recognition, it suffered from overfitting and lower generalization on real-time webcam input.

Our optimized architecture achieved performance comparable to or exceeding that of related studies [9][10], despite being trained on a smaller, personalized dataset. This highlights that even lightweight CNN architectures can deliver strong performance when properly tuned and applied to targeted data.

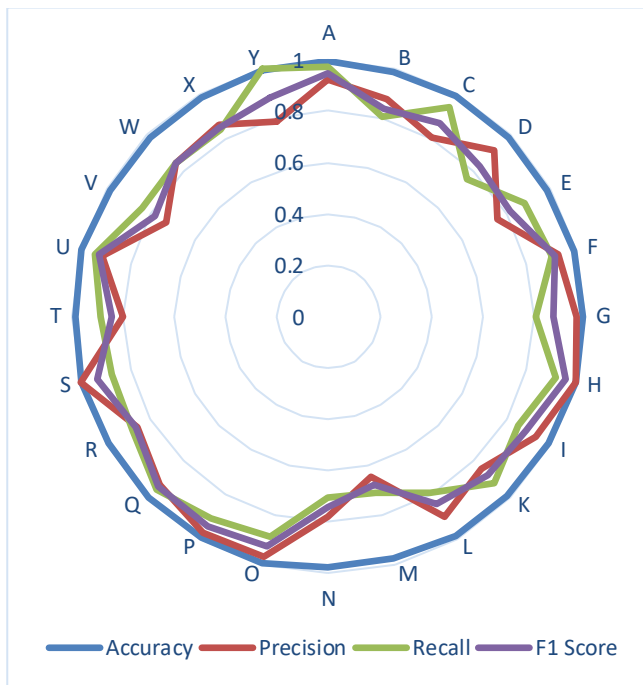


Fig. 18. Confusion matrix result.

Despite the promising results, this study has several limitations. The dataset used was relatively small and collected in controlled conditions, which may not fully capture variations in lighting, background, and hand positioning encountered in real-world environments. Furthermore, the scope of recognition was restricted to 24 SIBI alphabet signs, excluding letters J and Z due to their dynamic motion-based representation. These limitations may affect the model's generalizability when applied to more diverse or continuous sign language inputs.

Nevertheless, the findings demonstrate that lightweight CNN architectures can be effectively optimized to deliver robust performance even with modest datasets. This highlights the potential for deploying such models in practical applications, such as educational tools, real-time communication aids, and mobile platforms designed to bridge communication gaps for the deaf community.

Future research could focus on expanding the dataset to include more participants, environmental variations, and dynamic gestures such as J and Z. Integrating the model with real-time mobile or web-based applications would also provide practical validation of its effectiveness in everyday scenarios. Additionally, exploring advanced architectures, such as transfer learning or hybrid CNN-RNN models, may further improve recognition accuracy and support continuous sign language translation.

VI. CONCLUSION

This study demonstrates that optimizing the number of convolutional filters and dropout rates significantly improves the performance of CNN-based models for Indonesian Sign Language (SIBI) recognition. The best configuration achieved 87.1% accuracy on the test set and 90.25% accuracy in real-time scenarios, with no indication of overfitting or underfitting.

The results indicate that lightweight CNN architectures, when properly tuned, can perform effectively even in resource-constrained environments. Compared to heavier architectures like Faster R-CNN, YOLOv4, AlexNet, or VGG reported in other studies, the proposed model offers advantages in terms of computational efficiency and speed. However, a direct empirical comparison on the same dataset remains to be conducted to fully validate claims of superiority.

Currently, the system relies on cropped hand images, limiting its flexibility in real-world scenarios. Automatic hand detection through object detection methods has not yet been implemented, representing an important avenue for future improvement.

Future work will focus on extending the model to recognize dynamic gestures, such as the letters J and Z, integrating object detection techniques to automatically identify relevant hand regions, and deploying the system as a mobile-based application for real-time sign language interpretation. Incorporating these enhancements will improve robustness and usability, making the system more practical for inclusive communication tools.

ACKNOWLEDGMENT

This research was funded through a collaboration between the Department of Information Systems and the Department of Research and Community Services at Universitas Amikom Yogyakarta, Indonesia.

REFERENCES

- [1] World Health Organization, "Deafness and Hearing Loss," 2021. <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss> (accessed Jul. 09, 2022).
- [2] A. Harpini, "Disabilitas Rungu," Pusat Data dan Informasi Kementerian Kesehatan Republik Indonesia, 2019. <https://pusdatin.kemkes.go.id/resources/download/pusdatin/infodatin/infodatin-tunarungu-2019.pdf> (accessed Jul. 10, 2022).
- [3] S. T. Isma, "Meneliti Bahasa Isyarat dalam Perspektif Variasi Bahasa," 2018. [Online]. Available: http://kbi.kemdikbud.go.id/kbi_back/file/dokumen_makalah/dokumen_makalah_1540468871.pdf.
- [4] "Kamus SIBI," Kementerian Pendidikan dan Kebudayaan dan Lembaga Penelitian dan Pengembangan Sistem Isyarat Bahasa Indonesia, 2020. <https://pmpk.kemdikbud.go.id/sibi/> (accessed Jul. 14, 2022).
- [5] TensorFlow Team, "Distributed training with TensorFlow," TensorFlow, 2024. [Online]. Available: https://www.tensorflow.org/guide/distributed_training.
- [6] A. Deshmukh, "Real-Time Indian Sign Language Recognition Using CNNs for Communication Accessibility," *International Journal of Multidisciplinary Research and Analysis*, vol. 7, no. 9, pp. 4447-4453, 2024, doi: 10.47191/ijmra/v7-i09-41.
- [7] M. S. Arshad Malik, N. Kousar, T. Abdullah, M. Ahmed, F. Rasheed, and M. Awais, "Pakistan sign language detection using PCA and KNN," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 4, pp. 78-81, 2018, doi: 10.14569/IJACSA.2018.090414.
- [8] R. A. Alawwad, O. Bchir, and M. M. Ben Ismail, "Arabic Sign Language Recognition using Faster R-CNN," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 3, pp. 692-700, 2021, doi: 10.14569/IJACSA.2021.0120380.
- [9] D. Talukder and F. Jahara, "Real-Time Bangla Sign Language Detection with Sentence and Speech Generation," *ICCIT 2020 - 23rd Int. Conf. Comput. Inf. Technol. Proc.*, no. April, 2020, doi: 10.1109/ICCIT51783.2020.9392693.
- [10] J. Rocha, J. Lensk, T. Ferreira, and M. Ferreira, "Towards a Tool to Translate Brazilian Sign Language (Libras) to Brazilian Portuguese and Improve Communication with Deaf," *Proc. IEEE Symp. Vis. Lang.*

- Human-Centric Comput. VL/HCC, vol. 2020-Augus, no. July, 2020, doi: 10.1109/VL/HCC50065.2020.9127257.
- [11] S. Dwijayanti, Hermawati, S. I. Taqiyyah, H. Hikmarika, and B. Y. Suprpto, "Indonesia Sign Language Recognition using Convolutional Neural Network," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 10, pp. 415–422, 2021, doi: 10.14569/IJACSA.2021.0121046.
- [12] A. B. Rebrastya, "Indonesian Sign Language (SIBI) Dataset," 2022. <https://www.kaggle.com/datasets/alvinbintang/sibi-dataset>.
- [13] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, no. April 2018, pp. 1–6, 2018, doi: 10.1109/ICEngTechnol.2017.8308186.
- [14] V. Inacio, F. Carvalho, R. Rodrigues, dan A. Ferreira, "Statistical evaluation of medical tests," *arXiv*, 2020. [Online]. Available: <https://arxiv.org/abs/2007.07687>.
- [15] S. Tangirala, "Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 2, pp. 612–619, 2020, doi: 10.14569/ijacsa.2020.0110277.
- [16] Z. Mahmood, L. Jamel, D. A. Salem, and I. Ashraf, "Improving learning from the complex multi-class imbalanced and overlapped data by mapping into higher dimension using SVM++," *Scientific Reports*, vol. 15, no. 31245, 2025. [Online]. Available: <https://www.nature.com/articles/s41598-025-13929-w>.
- [17] T. Al-Shehari, M. Kadrie, M. N. Al-Mhiqani, T. Alfakih, H. Alsalman, M. Uddin, S. S. Ullah, and A. Dandoush, "Comparative evaluation of data imbalance addressing techniques for CNN-based insider threat detection," *Scientific Reports*, vol. 14, no. 24715, 2024. [Online]. Available: <https://www.nature.com/articles/s41598-024-73510-9>.
- [18] H. Li, G. K. Rajbahadur, D. Lin, C.-P. Bezemer, Z. Ming, and J. Jiang, "Keeping deep learning models in check: A history-based approach to mitigate overfitting," *arXiv*, 2024. [Online]. Available: <https://arxiv.org/abs/2401.10359>.