Jurnal Sisfotek Global
ISSN (Online): 2721 - 3161, ISSN (Print): 2088 – 1762
DOI: http://dx.doi.org/10.38101/sisfotek.v15i1.15678
Vol. 15, No. 1, March 2025, pp. 1-8

Introduction of Indonesian Significant Alphabet Images (BISINDO) using The Convolutional Neural Network Algorithm

Stefanus Kabut[1], Yoseph Pius Kurniawan Kelen[2], Budiman Baso[3], & Debora Chrisinta[4]

[1,2,3,4] Program Studi Teknologi Informasi, Universitas Timor, Sasi Timor Tengah Utara
email: stefanuskabut@gmail.com, yosepkelen@unimor.ac.id, budimanbaso@gmail.com, deborachrisinta@unimor.ac.id

ARTICLE HISTORY

Received : August 14th, 2024
Revised : September 9th, 2024
Accepted : September 26th, 2024

KEYWORDS

Alphabet
Bisindo
Convolutional Neural Network
API

ABSTRACT

Bisindo alphabet recognition is the process by which a computer system or software recognizes and recognizes the letters of the Bisindo alphabet. The Bisindo alphabet is a special alphabet used to communicate with people who are hearing or speech impaired. This process uses image processing and machine learning techniques to identify and classify each letter based on its shape and visual characteristics. This study used a dataset consisting of 520 Kaggle images divided into 26 categories. These images are resized, normalized and scaled up to improve model performance. A Convolutional Neural Network (CNN) model was developed and achieved 99.12587% accuracy after training. After the model was developed, the API was implemented using Flask. API functionality is tested using online interactions, ensuring accurate responses to image classification before implementation in mobile applications.

1. Introduction

According to statistics from the Indonesian Ministry of Health in 2019, around 18.9 million people in this country are deaf. However, most of them do not actively participate in the world of work. People with hearing impairments often feel that their ability to speak is impaired so they use sign language and body language to communicate. Sign language is a communication tool for deaf and speech-impaired individuals to interact with people around them. Deaf and speech-impaired individuals face difficulties communicating with people who are not hearing impaired.

A previous study conducted by [1], using the convolutional neural network method for classifying Indonesian sign language alphabets (Bisindo) obtained an accuracy of 52% of all letters. As for previous research conducted by [2], verification obtained 82%, 120 image distributions were not identified correctly and 880 images were identified correctly, this research compares two methods Support Vector Machine (SVM) and K-Nearest Neighbor (KNN).

Previous research conducted [3] using the convolutional neural network method provided an accuracy of 80.76%. The test involved 2 volunteers who carried out the test 52 times using the training ranking model. The results of the confusion matrix formula test show precision, recall, specificity and sensitivity values of 80.76% each. Further research was also carried out by [4] using Learning Vector Quantization (LVQ). The test results of this research were able to recognize 26 sign letters with an accuracy level of 61.54%. Next, carried out by [5], this research used the Support Vector Machine method to obtain the best results in this research with an accuracy level reaching 98.10%.

In this research, there are several problem formulations that will be answered, including how to determine the most suitable Convolutional Neural Network (CNN) architecture for recognizing the Indonesian Sign Language alphabet (BISINDO). This research also explores the ability of the CNN model to recognize Indonesian alphabets in real-time. With the right approach, it is hoped that CNN can be an effective solution in supporting communication for the community of people with hearing disabilities in Indonesia, especially in identifying BISINDO letters quickly and accurately

Based on the results of previous research and the problem formulation in this research, it can be concluded that the introduction of sign language has been mostly carried out using statistical data. This causes the classification process to take a long time. The convolutional neural network method is often used in previous research to carry out classification and has succeeded in achieving a fairly high level of accuracy.

This research aims to get to know Indonesian sign language, introduction to the letters of the alphabet A-Z. This research applies a convolutional neural network algorithm using a maxpoling convolution layer

2. Literature Review

2.1 Sign Language

According to the Big Indonesian Dictionary (KBBI), sign language is a language that does not use human speech sounds or writing in its symbol system. Sign language uses movements of fingers, hands, head, body, and so on, which are specifically created by deaf people and for deaf people (sometimes also for hearing people) [6]. Every country has its own unique sign language. Even in countries that speak the same language, such as the United States, sign languages remain different.

Humans interact with each other through communication in the form of language. Communication can occur both verbally and non-verbally [7]. In everyday life, humans communicate verbally but not all humans can communicate verbally

2.2 Machine Learning

According to Shalev-Shwartz and Ben-David, machine learning is the study of algorithms used to perform certain tasks that humans do automatically. As time passes, smart or intelligent machines will slowly replace and enhance human capabilities in various fields [8]. Machine learning is a part of artificial intelligence (AI) that helps computers or learning machines learn from past knowledge and make intelligent decisions [9]. Artificial Intelligence is a field in computer science that is aimed at creating software and hardware that can function as something that can think like humans [10].

2.3 CNN Method

Convolutional Neural Network (CNN) algorithms are very popular in deep learning due to their ability to extract features that can be tailored to specific tasks, enabling the recognition of new objects within existing networks [11]. CNNs consist of neurons with weights, biases, and activation functions. The convolution layer in a CNN forms a filter with a certain length and height. Like other neural networks, CNN consists of input layers, output layers, and several hidden layers [12]. These layers perform operations to modify data with the aim of learning special features of that data. The three most common layers in a CNN are convolution layers, activation function or ReLU, and pooling. Convolution layers apply a series of filters to the input image, where each filter functions to activate certain features of the image**.**

3. Research Method

There are several steps used for modeling development starting from data collection, data argumentation, data preprocessing, data slipping, model creation, testing, to saving the model in H5 and Json format. The model and model presentation that can be observed in Figure 1 are as:

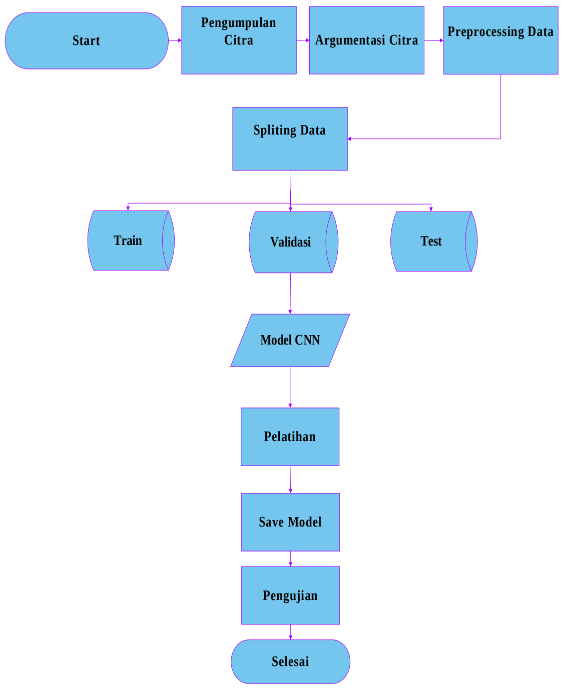


Figure 1. Modeling Development Steps

3.1 Data collection

At this stage the author and his team conducted a literature study taking data on Kaggle and sourced the dataset from [13],[14]and [15]. The dataset he took was image data of hand movements of the Indonesian language alphabet (BISINDO).

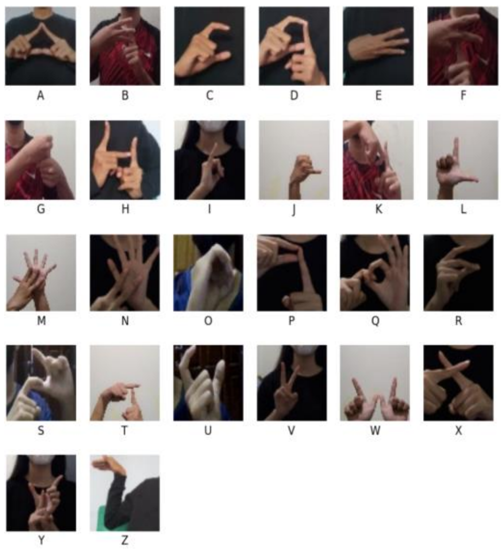


Figure 2. Sample image consisting of letters A-Z

Images taken on Kaggle, obtained five images for each different background, which means 20 images for each alphabet. The total images obtained from the letters A to Z were 520 images.

3.2 Image Augmentation

The data augmentation process was carried out by multiplying the original 20 images from each class (20 images for each alphabet) to 880 images for each letter.

The image transformation used is brightness, image translation, zoom, and image rotation to rotate 64 x 64 pixels. The following is an image of the results of data augmentation.

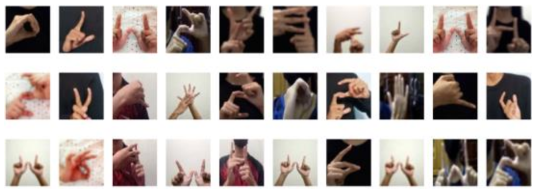


Figure 3. Data Augmentation Results

3.3 Splitting data

The Spliting process is divided into three main parts, namely Training data, Validation and Testing. It can be seen in table 1 below:

Table 1. Splitting Data

No	*Datasets*	*Split*	*Amount*
1	Traning	80	704
2	Validation	20	88
3	Testing	20	88

3.4 Data preprocessing

To process the dataset before model training, the image_dataset_from_directory utility from TensorFlow is used. The training dataset consists of 704 files, the validation dataset consists of 88 files, and the testing dataset consists of 88 files, all divided into 26 different classes. Batch size and image s**ize are set using the variables** BATCH_SIZE and RESIZED_IMAGE_SIZE to ensure consistency in data processing. The settings shuffle=True and seed=1 are applied so that the dataset is shuffled and consistent results are obtained every time this process is run.

3.5 CNN Models

To increase model accuracy, this research uses the Adaptive Moment Estimation (ADAM) optimizer. Adaptive Moment Estimation combines RMSprop and Stochastic Gradient Descent with momentum, so it can handle gradient descent more effectively and efficiently. This optimizer uses the first (mean) and second (variance) moments of the gradient to perform parameter updates, allowing adaptive adjustment of the learning rate during the training process.

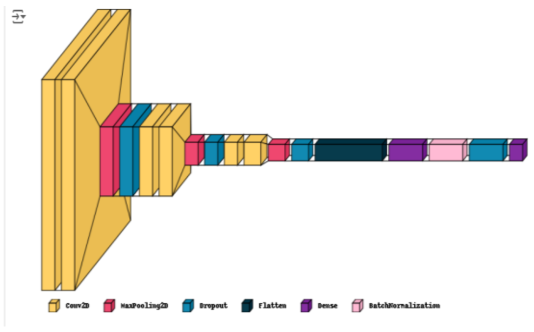


Figure 4. CNN Models Architecture

The application of the convolutional neural network (CNN) architectural model is applied to the BISINDO image data training process to increase recognition accuracy. The input data consists of a three-color image, namely Red, Green, Blue (RGB) measuring 64x64.

Table 2. CNN architecture

Layer Type	*Output Shape*	*Param*	*Description*
conv2d_27 (Conv2D)	(None, 64, 64, 16)	208	First convolutional layer, 16 filters
max_pooling2d_18 (MaxPooling2D)	(None, 32, 32, 32)	0	First max pooling layer
dropout_12 (Dropout)	(None, 32, 32, 32)	0	First dropout layer
conv2d_31 (Conv2D)	(None, 16, 16, 128)	32896	Fifth convolutional layer, 128 filters
max_pooling2d_20 (MaxPooling2D)	(None, 8, 8, 256)	0	Third max pooling layer
flatten_6 (Flatten)	(None, 16384)	0	Flattening layer
dense_12 (Dense)	(None, 512)	8389120	First dense layer with 512 units
dense_13 (Dense)	(None, 26)	13338	Output layer with 26 units

3.6 Data Training

This model was developed with an architecture that has been selected based on the evaluation carried out in the previous sub-sections. The model structure includes six convolution layers and three maxpool layers with a pooling kernel size of 3x3. For optimization, this model uses RMSprop and is trained for 50 epochs. The image received by the model has a resolution of 64 x 64 x 3, which means the image is 64 x 64 pixels in size with three channels each representing the colors Red, Green, and Blue (RGB).

3.7 Testing

Below is a brief explanation of how images are classified by model. First, the 64x64x3 image enters the input layer and is passed to the first convolution using 16 filters and the ReLU activation function. The image then goes through a second convolution with 32 filters and a ReLU activation function. After the image passes through two convolutional layers, it is processed by maxpooling with a 3x3 kernel and goes to the dropout layer with a dropout rate of 25%. The image then goes through a third convolution with 32 filters and a ReLU activation function, then a fourth convolution with 64 filters and a ReLU activation function. After passing through a total of four convolutional layers, the image is reprocessed with 3x3 maxpooling and layer dropout with a dropout rate of 25%. The process continues with the fifth convolution with filter 128 and ReLU activation function, then the sixth convolution with filter 256 and ReLU activation function. After passing through a total of 6 convolutional layers, the image is processed with 3x3 maxpooling and enters the dropout layer with a dropout rate of 25%. Finally, the image is passed to the first dense layer with 512 filters and ReLU activation function.

4. Result and Discussion

The steps taken in the CNN model are carrying out alphabet recognition analysis. By entering a test image whose object is traced according to the alphabet entered from the test data, the results will be displayed as in the following image.

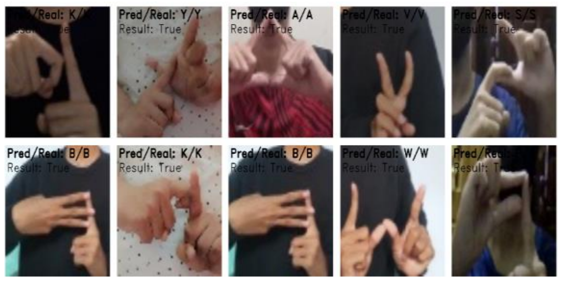
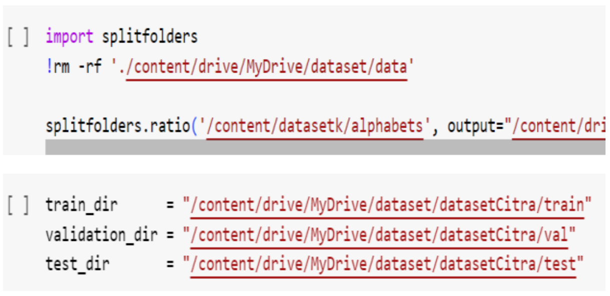


Figure 5. Image test results

4.1 Splitting Data

Training data is stored in the train folder, validation data is stored in the val folder, and testing data is stored in the test folder. The training data contains 704 images, the testing data contains 88 images and the validation data also contains 88 images. The author uses a comparison ratio of 80: 20: 20.

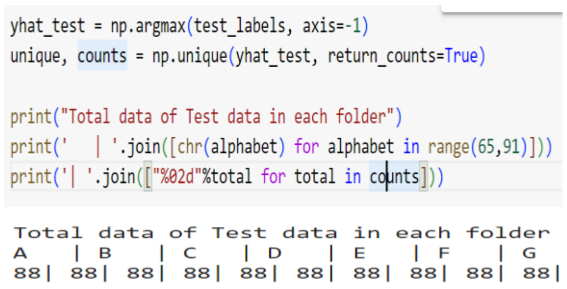


```
[ ] import splitfolders
    !rm -rf './content/drive/MyDrive/dataset/data'

    splitfolders.ratio('/content/datasetk/alphabets', output="/content/dri
```

```
[ ] train_dir      = "/content/drive/MyDrive/dataset/datasetCitra/train"
    validation_dir = "/content/drive/MyDrive/dataset/datasetCitra/val"
    test_dir       = "/content/drive/MyDrive/dataset/datasetCitra/test"
```

Figure 6. Source code Splitting data

The np.argmax function converts test_labels from a one-hot encoded format to an array of integer class labels. Next, np.unique counts the number of samples in each class, producing two arrays: unique with the class label's unique values and counts with the number of occurrences of each class. The table header is printed by concatenating the letters A through Z using the | separator, followed by printing the number of test data in each class formatted into a two-digit string and concatenated with the same separator



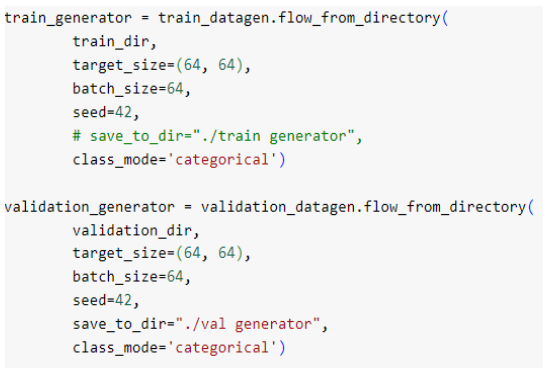
```
yhat_test = np.argmax(test_labels, axis=-1)
unique, counts = np.unique(yhat_test, return_counts=True)

print("Total data of Test data in each folder")
print('   | '.join([chr(alphabet) for alphabet in range(65,91)]))
print('| '.join(["%02d"%total for total in counts]))
```

```
Total data of Test data in each folder
A    | B    | C    | D    | E    | F    | G
88| 88| 88| 88| 88| 88| 88| 88| 88| 88|
```

Figure 7. Source Displays Testing data

4.2 Preprocesing Data

At this stage, the image data that has been augmented amounts to 22906 images consisting of 26 classes. The author indexes the data for each class and the class format starts from index 0 to 25. The following is a picture of indexing image data labels



```
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(64, 64),
        batch_size=64,
        seed=42,
        # save_to_dir="./train generator",
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        validation_dir,
        target_size=(64, 64),
        batch_size=64,
        seed=42,
        save_to_dir="./val generator",
        class_mode='categorical')
```

Figure 8. Source data preprocessing code

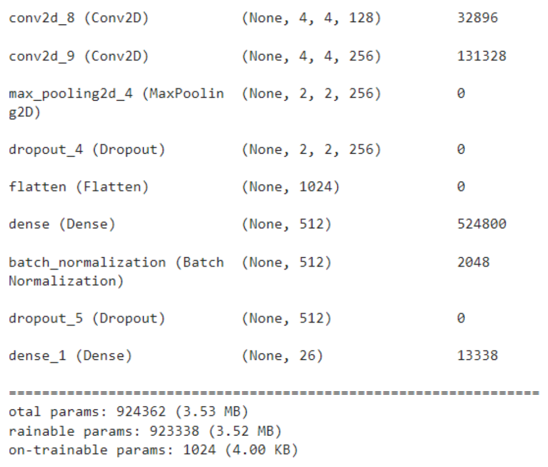
In the validation data set, the percentage of data per class varied slightly, ranging from 3.63% to 4.02%. Class Z has the highest percentage, namely 4.02%, while classes S and W have the lowest percentage, namely 3.63%.

Table 3. Percentage of Train and Validation data

Index	*Class*	*Train Percentage*	*Validation Percentage*
0	A	3.85%	3.98%
1	B	3.85%	3.98%
2	C	3.85%	3.80%
3	D	3.85%	3.85%
4	E	3.85%	3.76%
5	F	3.85%	3.98%
6	G	3.85%	3.89%
7	H	3.85%	3.93%
8	I	3.85%	3.80%
9	J	3.85%	3.80%
10	K	3.85%	3.89%
11	L	3.85%	3.67%
12	M	3.85%	3.93%
13	N	3.85%	3.80%
14	O	3.85%	3.89%
15	P	3.85%	3.98%
16	Q	3.85%	3.80%
17	R	3.85%	3.93%
18	S	3.85%	3.63%
19	T	3.85%	3.89%
20	U	3.85%	3.72%
21	V	3.85%	3.89%
22	W	3.85%	3.63%
23	X	3.85%	3.85%
24	Y	3.85%	3.72%
25	Z	3.85%	4.02%

4.3 Models CNN

The author uses epoch 50 in making the model. The following are the results of the model that has been run:



```
conv2d_8 (Conv2D)           (None, 4, 4, 128)         32896

conv2d_9 (Conv2D)           (None, 4, 4, 256)         131328

max_pooling2d_4 (MaxPoolin  (None, 2, 2, 256)         0
g2D)

dropout_4 (Dropout)         (None, 2, 2, 256)         0

flatten (Flatten)           (None, 1024)              0

dense (Dense)               (None, 512)               524800

batch_normalization (Batch  (None, 512)               2048
Normalization)

dropout_5 (Dropout)         (None, 512)               0

dense_1 (Dense)             (None, 26)                13338

=================================================================
otal params: 924362 (3.53 MB)
rainable params: 923338 (3.52 MB)
on-trainable params: 1024 (4.00 KB)
```

Figure 9. Output from the sequential CNN model

The CNN architecture used consists of six convolution layers and three maxpool layers with a 3x3 pooling kernel. This model will go through training for 50 epochs. Next, the model will be compiled using the RMSprop optimizer and using the categorical crossentropy loss function for classification purposes. To monitor training performance, history callbacks will also be included in the model. Each change in epoch values, model optimizer choices, and CNN architecture configuration will be explained separately to provide a clearer picture. The results of changing the epoch value, using the optimizer model, and CNN architectural configuration will be explained respectively.

4.4 Number of Epochs

The author uses epochs 5 times, consisting of 100 epochs, 50 epochs, 30 epochs, 15 epochs and 10 epochs. Comparison of model performance by number of epochs.

Table 4. Comparison of model performance by number of epochs

Epochs Total	*Loss*	*Accuracy*
100	0.0514889	99.21328%
50	0.1347526	99.12587%
30	0.3879700	90.07867%
15	0.2936137	99.78147%
10	0.2045909	99.78147%

4.5 Number of Layers

From the table, it can be concluded that increasing or decreasing the number of convolution or pooling layers does not always improve model performance. Reducing the dimensions of the pooling kernel to increase the number of parameters does not always improve the model accuracy. Overall, finding the optimal model architecture requires an iterative approach of trial and error.

Table 5. Number of layers and model performance

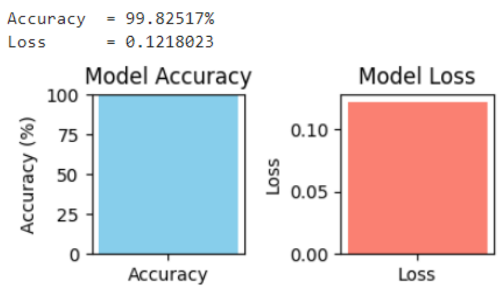
Number of Layers Conv2D	*Number of Layers MaxPool*	*Kernel Size*	*Trainable Params*
2	2	3x3	941018
4	4	2x2	924362
6	6	2x2	3547338
8	8	3x3	8583402

4.6 Model Training

The image processing pipeline starts with an input image of 64 x 64 x 3. The image goes through two convolution layers (16 filters and 32 filters, both with ReLU), then 3x3 maxpooling and 25% dropout. This process is repeated with additional convolutions (32 filters, 64 filters, 128 filters, and 256 filters) followed by 3x3 maxpooling and 25% dropout. After that, the image is processed through a dense layer with 512 units and ReLU, normalized with batch normalization (0.99 momentum), and 50% dropout. Finally, the image is classified in the second dense layer with 26 units and sigmoid. This process is repeated during training to improve the accuracy and generalization of the model.

4.7 Loss Score and Accuracy

With six convolution layers and three max pooling layers with a pooling kernel size of 3x3. Apart from the training duration which can be said to be relatively fast, the choice of architecture is also based on the smallest loss value and the highest accuracy value compared to the other three options.

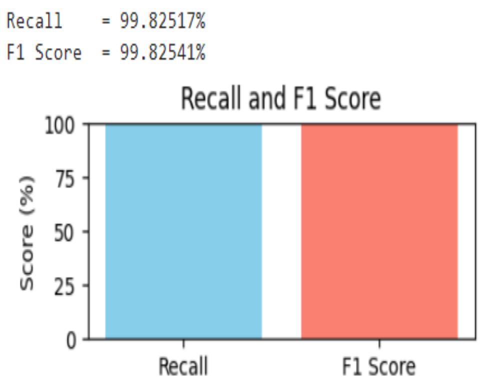


```
Accuracy   = 99.82517%
Loss       = 0.1218023
```

Figure 9. Loss and accuracy score output

4.8 Recall and F1 Score

After predicting the test data, the recall score was calculated using the recall_score function from the sklearn library, resulting in a value of 99.82517%. The F1 score calculated with the f1_score function shows a value of 99.82541%. The following figure illustrates the use of these two functions in model performance evaluation.



```
Recall    = 99.82517%
F1 Score  = 99.82541%
```

Figure 10. Recall dan F1 Score

4.9 Confidence Value

The table below shows the incorrect predictions from the model, with each row showing the true label and the model's confidence percentage. All predictions have almost the same level of confidence, ranging from 21.82% to 21.92%. This similarity indicates that the model has high uncertainty in distinguishing between different classes, which may indicate problems in model training or data quality. This consistent and low confidence value indicates that the model has difficulty providing accurate predictions.

Table 6. Confidence Value

Actual	*Confidence* (%)
a	21.84%
b	21.87%
c	21.83%
d	21.83%
e	21.85%
f	21.85%
g	21.84%
h	21.82%
i	21.87%
j	21.85%
k	21.84%
l	21.86%
m	21.88%
n	21.87%
p	21.84%
q	21.92%
r	21.82%
s	21.87%
t	21.82%
u	21.90%
v	21.83%
w	21.87%
x	21.87%
y	21.84%
z	21.83%

4.10 Data Testing

Testing this model consists of four main stages of using the model in .h5 format, evaluating with data testing, testing via a web browser, and testing the API with Postman. The .h5 file is the result of training a CNN model to recognize image patterns.

The testing process begins by using a model in .h5 format, the result of CNN training which develops the model's ability to recognize image patterns and features.

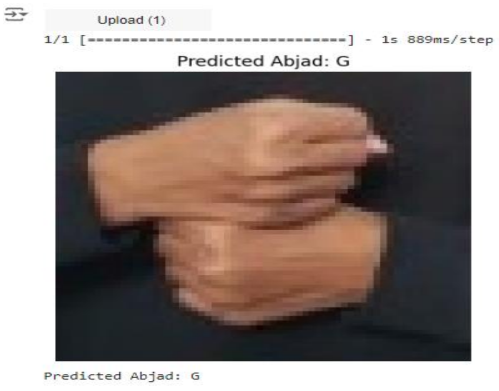

```
Upload (1)
1/1 [==============================] - 1s 889ms/step
Predicted Abjad: G
Predicted Abjad: G
```

Figure 11. Testing Using Model.h5

In the second option, the test data is loaded from the folder into the array, as are the labels. Ten images were selected randomly and processed with the predict function. The prediction results are compared with the labels, returning True if they match and False if they don't. The following image shows that all predictions are correct

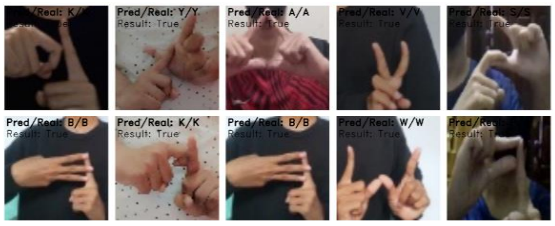


Figure 12. Testing Using testing data

The third option involves testing the model with data via a web browser. The author saves the model in .h5 format, then tests using a web browser after creating code with Python, JavaScript, and an interactive display with HTML and CSS. Stored models are loaded and called via API, enabling seamless integration and interaction with web applications.

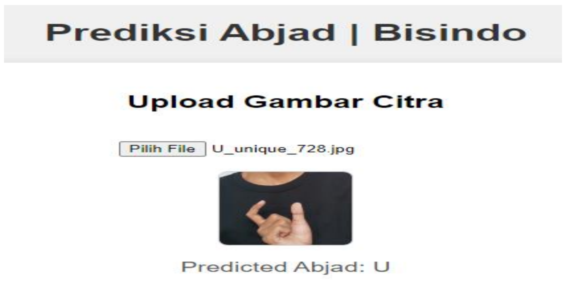
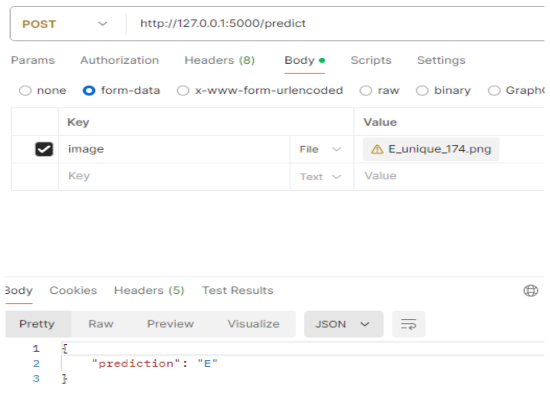


Figure 13. Testing using a web browser

The fourth option involves testing the model with data via Postman. The author saved the model in .h5 format and tested the model using Postman after coding in Python and JavaScript. Stored models are loaded and called via API, allowing testing and direct interaction with models via Postman.



```
{
    "prediction": "E"
}
```

Figure 14. Testing Using Postman

5. Conclusion

Based on the results of the Convolutional Neural Network (CNN) model training that has been carried out, it can be concluded that the CNN model in this research, after going through 3 convolution layers and 3 max pooling layers, has a configuration with an input shape of 64x64 pixels and a filter measuring 3x3. The training process lasted for 50 epochs with 744 training data, 88 images for validation data, and 88 images for testing data. This model succeeded in achieving high accuracy of 99.21328% in recognizing Indonesian Sign Language (BISINDO) alphabet images. Apart from that, the recall score obtained from data testing reached 99.82517%, and the f1_score value calculated using the Sklearn library reached 99.82541%. Testing was carried out using four different methods, namely data testing, the h5 model which was tested via Google Colaboratory, web browser, and Postman.

This research has several weaknesses, one of which is the inability to determine optimal parameters, which requires the use of trial and error methods to achieve a high level of accuracy. However, this research has the potential to be developed further, such as adding number recognition in sign language, as well as developing it into an application that can be integrated with mobile devices.

References

[1] A. Hibatullah and I. Maliki, “Penerapan Metode Convolutional Neural Network,” *Unikom*, pp. 1–8, 2019.

[2] C. Umam and L. B. Handoko, “Convolutional Neural Network (CNN) Untuk Identifkasi Karakter Hiragana,” *Pros. Semin. Nas. Lppm Ump*, vol. 0, no. 0, pp. 527–533, 2020, [Online]. Available: https://semnaslppm.ump.ac.id/index.php/semnaslppm/article/view/199

[3] M. Sholawati, K. Auliasari, and F. Ariwibisono, “Pengembangan Aplikasi Pengenalan Bahasa Isyarat Abjad Sibi Menggunakan Metode Convolutional Neural

Network (Cnn),” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 6, no. 1, pp. 134–144, 2022, doi: 10.36040/jati.v6i1.4507.

[4] S. R. Yulian and S. Suhartono, “Pengenalan Bahasa Isyarat Huruf Abjad Menggunakan Metode Learning Vector Quantization (LVQ),” *J. Masy. Inform.*, vol. 8, no. 1, pp. 1–8, 2017, doi: 10.14710/jmasif.8.1.31450.

[5] Y. P. K. Kelen and B. Baso, “Klasifikasi Tenun Timor Menggunakan Metode SVM Berdasarkan Speeded Up Robust Features,” *J. Teknol. Inf. dan Ilmu Komput.*, vol. 10, no. 6, pp. 1353–1360, 2023, doi: 10.25126/jtiik.1067625.

[6] O. Mailani, I. Nuraeni, S. A. Syakila, and J. Lazuardi, “Bahasa Sebagai Alat Komunikasi Dalam Kehidupan Manusia,” *Kampret J.*, vol. 1, no. 1, pp. 1–10, 2022, doi: 10.35335/kampret.v1i1.8.

[7] C. Li, S. Xiuting, W. Yan, T. Development, and I. Technology, “陈 黎 1， 盛秀婷 2， 吴岩 3 (1.,” vol. 4, no. 19, pp. 8–14, 2022.

[8] A. Roihan, P. A. Sunarya, and A. S. Rafika, “Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper,” *IJCIT (Indonesian J. Comput. Inf. Technol.*, vol. 5, no. 1, pp. 75–82, 2020, doi: 10.31294/ijcit.v5i1.7951.

[9] A. Fathurohman, “Machine Learning Untuk Pendidikan: Mengapa Dan Bagaimana,” *J. Inform. dan Teknol. Komput.*, vol. 1, no. 3, pp. 57–62, 2021.

[10] A. Sunarya, S. Santoso, and W. Sentanu, “Sistem Pakar Untuk Mendiagnosa Gangguan Jaringan Lan,” *Creat. Commun. Innov. Technol. J.*, vol. 8, no. 2, pp. 1–11, 2015.

[11] Verdy and Ery Hartati, “Klasifikasi Penyakit Mata Menggunakan Convolutional Neural Network Model Resnet-50,” *J. Rekayasa Sist. Inf. dan Teknol.*, vol. 1, no. 3, pp. 199–206, 2024, doi: 10.59407/jrsit.v1i3.529.

[12] I. Suhardin, A. Patombongi, and A. M. Islah, “MENGIDENTIFIKASI JENIS TANAMAN BERDASARKAN CITRA DAUN MENGGUNAKAN AlGORITMA CONVOLUTIONAL NEURAL NETWORK,” *Simtek J. Sist. Inf. dan Tek. Komput.*, vol. 6, no. 2, pp. 100–108, 2021, doi: 10.51876/simtek.v6i2.101.

[13] Alfredolorentiars, “bisindo leter dataset.” https://www.kaggle.com/alfredolorentiars/datasets

[14] Achmadnoer, “Bahasa Isyarat Indonesia (BISINDO) Alphabets”, [Online]. Available: https://www.kaggle.com/datasets/achmadnoer/alfabet-bisindo

[15] Idhamozi, “Dataset berupa foto Bahasa Isyarat Indonesia. Indonesian Sign Language BISINDO.” https://www.kaggle.com/datasets/idhamozi/indonesian-sign-language-bisindo