Journal of Applied and Research Computer Science and Information Systems
Vol. 1, No. 2, Desember2023, pp. 86~99
ISSN: 2988-294X-, DOI: 10.61098/jarcis.v1i2.87

Development of An Application Transforming Handwriting into Digital Form using CNN

Claudio Josulin[1], Yulia Ery Kurniawati[2]
[1,2] Informatics, Faculty of Computer Science and Design, Institut Teknologi dan Bisnis Kalbis
yuliaeryk@gmail.com

Article Info

Article history:

Received 12 01, 2023
Revised 12 15, 2023
Accepted 12 29, 2023

Keywords:

Computer vision
Convolutional Neural Network
Handwriting recognition
Handwriting Classification
ResNet50

ABSTRACT

With the increasing demand for efficient digitalization and the relevance of handwriting communication, this study aims to develop an application to recognize and predict handwriting using a Convolutional Neural Network (CNN) with ResNet50 architecture. The software development life cycle (SDLC) is an incremental model with two increments. The first increment is used to build the model, and the second increment is used to build the user interface. In the first increment, the data used in this study is handwritten images of Latin uppercase, Latin lowercase, and Arabian numerals with 62 classes. The training data used English Handwritten Characters by Dhruvil Dave from Kaggle Dataset. Data was trained and validated using k-fold cross-validation with tenfold and ten epochs for each fold. The model has accuracy, precision, recall, and f1-score of 66.33%, 73.4%, 66.2%, and 66%, respectively. The second increment was used to develop the application. The application was developed using Python 3.8 programming language with Tkinter, PIL, Tensorflow, Keras, and OpenCV libraries. The functional application can work as expected based on the black box testing. The developed application can predict handwriting with up to 50% accuracy.

Corresponding Author:

Yulia Ery Kurniawati
Institut Teknologi dan Bisnis Kalbis, Jakarta, Indonesia
Email: yuliaeryk@gmail.com

1. INTRODUCTION

The subject of image processing and pattern recognition is receiving a lot of attention these days, ranging from handwriting, facial patterns, fingerprints, and printed character pattern identification. Because of this, scientists need to enhance computer's ability to recognize and represent patterns [1]. Handwriting patterns are unique to each person. It is natural for people to have handwriting that is challenging to read because each person writes differently. The hard-to-read handwriting makes it difficult to carry out regular tasks, including transactions and administrative work.

There are studies that recognize the handwriting in other languages that have special characters such as Bangla[2], Arabic [3], [4]Thai [5], [6]. Some studies also developed applications to help read handwriting in different characters such as Android based applications for teaching aid for kindergarten to primary school level students to help them practicing their handwriting using CNN [7] and an Android based application to recognize text or characters using CNN and BiLSTM model for the text recognition [8].

Convolutional Neural Network (CNN) is a popular algorithm for pattern recognitio [9], [10], [11], [12], [13], [14]. CNN uses images as input. Then, the images will be extracted to get features and reduce the size without altering their quality [15]. Because of this, CNN has gained popularity and achieved positive outcomes [16]. In the study "Comparison of the Performance of CNN LeNet5 and Extreme Learning Machine in Handwritten Image Recognition of Numbers", handwritten patterns of numbers were recognized with an

accuracy of 98.04% using the CNN combined with the LeNet5 architecture [17]. In addition, CNN is frequently used to identify writing in foreign scripts. The study “Recognition of Arabic Script Handwriting Patterns Using the Convolution Neural Network Method” yielded the best accuracy rate of 78.10% in handwriting recognition on Arabic script [16]. Moreover, research entitled "Convolutional Neural Network (CNN) for Hiragana Character Identification" to recognize handwriting in the Japanese Hiragana script obtained an accuracy of 82% [18].

Convolutional Neural Network (CNN) algorithms have various architectures, including LeNet, AlexNet, VGGNet (Visual Geometry Group), GoogLeNet (Inception), ResNet (Residual Network), and others [19]. The ResNet architecture reduces the level of difficulty in training, which results in increased performance in training and generalization error [20]. Jayakanthan et al., in research entitled “Handwritten Tamil Character Recognition Using ResNet,” proved that the ResNet architecture succeeded in obtaining the highest accuracy with an accuracy of 96.014% compared to the VGG-16 architecture, GoogLeNet, as well as the SVM (Support Vector Machine) algorithm, and Gabor Filters with SVM [21]. Atliha and Šešok compared the VGGNet architecture with ResNet for creating image captions in a study entitled "Comparison of VGG and ResNet used as Encoders for Image Captioning” [22]. It proves that the ResNet architecture achieved higher accuracy and lower loss than VGGNet in the training stage and validation.

This research will develop an application for recognizing handwritten images and transforming them into digital text. The CNN algorithm used to build the model because CNN is a better success than other algorithms and and used to solve of most latin languages. The Incremental Software Development Life Cycle (SDLC) method used to develop the application and used the Python 3.10 as the programming language. The Incremental method was chosen because the application development will be carried out in stages until it meets the user's needs [23].

2. METHOD

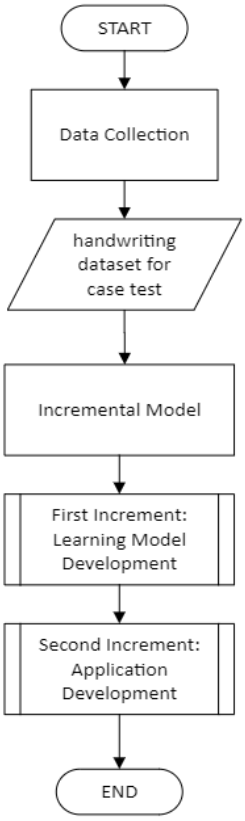


Figure 1. Research Flow

Figure 1 shows the research flow that used in this study. It started with data collecting. After the data has been collected, the next step is application development. The SDLC used in this study is the Incremental Model. It is divided into two increments. The first increment is learning model development. The second increment is application development.

2.1. Data Collection

Figure 2. English Handwritten Characters Dataset Samples

The primary data in this study is handwritten. The handwritten data taken from Kaggle.com is named English Handwritten Character by Dhruvil Dave. It is used to train the model. It has 3410 images that contain uppercase, lowercase, and Arabic numbers [24]. Training used k-fold cross-validation with ten folds. K-fold cross-validation divides the data into k folds, each used as testing data and the rest used as training data [25]. Figure 2 shows the samples of handwritten images in the English Handwritten Characters Dataset.

The case test data was handwritten by 16 people. They are asked to write "the quick brown fox jumps over a lazy dog 0123456789". Writing is done by randomizing capital and lowercase letters and randomizing numbers. Writing sentences will use a pen with blue ink, as shown in Figure 3. The handwriting will be filtered first to remove noise. After filtering, handwriting will be segmented per letter to be used as case data.

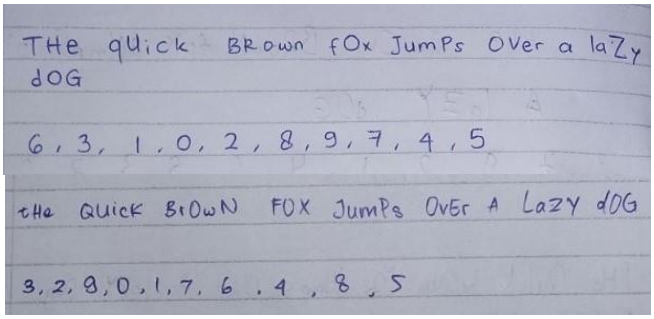


Figure 3. Case Test Data

2.2. Incremental Model

This study used the Incremental Model as SDLC to develop the application. The incremental model divides the requirements into multiple standalone modules [26]. Each increment consists of analysis, design, code, and test, as shown in Figure 4. In this study, two increments. The first applies the model's logic and trains the model with the dataset. Its process consists of system requirements analysis, model logic design, design implementation, and prediction model testing. The second one is developing an application through requirements analysis, the application design, implementation of the design, and testing the application.

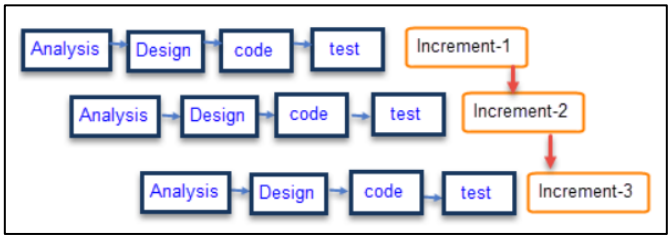


Figure 4. Case Test Data [27]

2.2.1. First Increment: Learning Model Development

The first increment is used to develop the learning model. In this stage, the features were extracted from the English Handwritten Character dataset by Dhruvil Dave. It is divided into four steps: analysis, design, code, and test. The analysis steps are used to analyze the requirements of the learning model. The next step is design, it used to design the flow of learning model development. The third step is code, it used to implement the design using Python programming language. The last stage is the test, which is used to evaluate the learning model using accuracy, precision, recall, and f1-score.

2.2.2. Second Increment: Application Development

The second increment is used to develop the application. This stage is divided into four steps: analysis, design, code, and test. The analysis step is used to describe the requirements to develop the application. Then

the design step is used to design the user interface application. It is implemented in the code steps into an application. In the last step or test, the application test uses black box testing to test the application functionalities.

3. RESULTS AND DISCUSSION

This study used an incremental model that was divided into two increments. The first increment was used to develop the training model, and the second was used to develop the application.

3.1. Increment I

The first increment is used to develop the training model. It is divided into four steps: analysis, design, implementation, and testing.

3.1.1 Analysis

This study began with collecting handwritten image data in .png format. Figure 5 shows the labels in the .csv file, so this research is supervised learning. In the dataset, there are ten numbers (0-9), 26 capital letters (A-Z), and 26 lowercase letters (a-z), so there are a total of 62 labels in the dataset. Each character has 55 images with a size of 1200x900 pixels.

Feature extraction was done using the ResNet algorithm with 50 convolution layers or ResNet50. Figure 5 shows the ResNet with 50 convolution layers or ResNet50. The feature extraction results will be classified into 62 classes. The entire feature extraction process is carried out using the Keras library. Meanwhile, the model evaluation will be carried out using the Scikit-Learn library.

In theory, the more learning layers, the better the quality of the resulting model will be as long as over-fitting can be overcome. However, many architectures suffer from the problem of "vanishing gradients" [28]. The vanishing gradients problem is a problem experienced by algorithms that update weights using a gradient basis, such as back-propagation, where the more learning layers used, the harder it will be for the model to identify light problems [29], [30]. The ResNet architecture is designed to solve this problem.

image	label
Img/img001-001.png	0
Img/img001-002.png	0
Img/img001-003.png	0
Img/img001-004.png	0
Img/img001-005.png	0
Img/img001-006.png	0
Img/img001-007.png	0
Img/img001-008.png	0
Img/img001-009.png	0
Img/img001-010.png	0
Img/img001-011.png	0
Img/img001-012.png	0
Img/img001-013.png	0
Img/img001-014.png	0

Figure 5. Labeled Dataset

ResNet guarantees that increasing learning layers will not produce worse accuracy than the algorithm with fewer layers [28] He et al., as the pioneer of the ResNet method, compared the conventional neural network (plains) method with ResNet, which uses 18 layers and 34 layers on the ImageNet 2012 dataset, respectively. The plains-18 method produces an error percentage of 27.94%, while the plains-34 method produces an error percentage of 28.54%. Meanwhile, the ResNet-18 and ResNet-34 methods have an error percentage of 27.88% and 25.03%, respectively[31]. Increasing the learning layer will not reduce the model's accuracy.

3.1.2. Design

This step will focus on the architectural design of the process to be carried out. Before processing the image, it will be resized to 224x224 pixels. The image format was originally black and white (grayscale) and was changed to RGB. The ResNet architecture can only receive images with three color channels. Because the existing labels are not all in number format (integer or float), one-hot encoding will be carried out, replacing labels in string form with array form with many columns that match the previous label. The array will then be made into an integer form whose encoded results are as in Table 1.

Table 1. Encode Label

Label	Encode
0	0.
1	1.
2	2.
3	3.
4	4.
5	5.
6	6.
7	7.
8	8.
9	9.
A	10.
B	11.
C	12.
D	13.
E	14.
F	15.
G	16.
H	17.
I	18.
J	19.
K	20.

Label	Encode
.L	21.
M	22.
N	23.
O	24.
P	25.
Q	26.
R	27.
S	28.
T	29.
U	30.
V	31.
W	32.
X	33.
Y	34.
Z	35.
a	36.
b	37.
c	38.
d	39.
e	40.
f	41.

Label	Encode
g	42.
h	43.
i	44.
j	45.
k	46.
l	47.
m	48.
n	49.
o	50.
p	51.
q	52.
r	53.
s	54.
t	55.
y	56.
v	57.
w	58.
x	59.
y	60.
z	61.

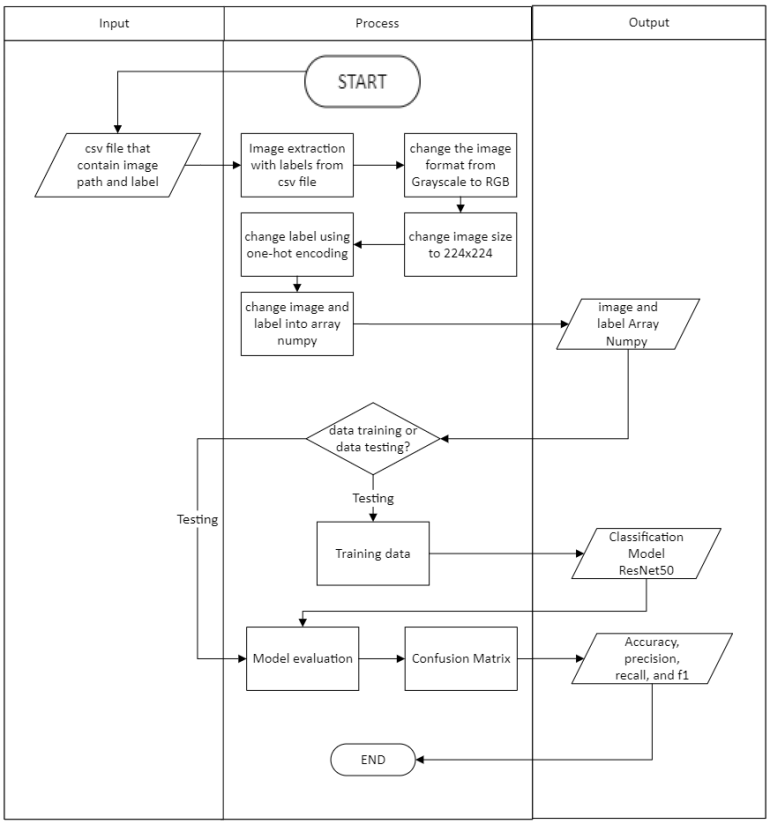


Figure 6. Flowchart of Increment I Process

The model in this study used the CNN algorithm with the ResNet-50 architecture, which has 50 convolution layers for feature extraction. Meanwhile, the weights that will be used are pre-trained weights from ImageNet. Data validation will use k-fold cross-validation with ten-fold. Model training on each fold will use ten epochs. The resulting model will be tested using data testing, which will use a confusion matrix. Figure 6 is the flowchart of the Incremental I process.

3.2.3. Implementation

In the implementation stage, the process of applying the design made into code will be carried out. The model will be created using the Python 3.8 programming language. The model-building process will use various Python libraries, including the OpenCV library for image data processing, the Keras library for building

feature extraction and class classification, and the Scikit-Learn library for one-hot encoding, k-fold cross-validation, and confusion matrix.

3.2.3. Test

The testing stage used a confusion matrix to test the model using testing data. The confusion matrix is square (dimensional nxn), where the rows represent the actual groups (labels) and the columns represent the groups (labels) predicted by the model [32]. In binary classification, the confusion matrix dimension 2x2 matrix that reports the number of True Positives (TP) or the number of 1 label that is predicted to be 1, True Negatives (TN) or the number of 0 labels that are predicted to be 0, False Positives (FP) or the number of 0 labels that are predicted to be 1, and False Negative (FN) or the number of labels 1 predicted 0.

$$\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} \quad (1)$$

Equation 1 is the confusion matrix. The confusion matrix can calculate accuracy, precision, recall, and F1-score. Equations 2, 3, 4, and 5 are the equations to calculate accuracy, precision, recall, and F1-score, respectively.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

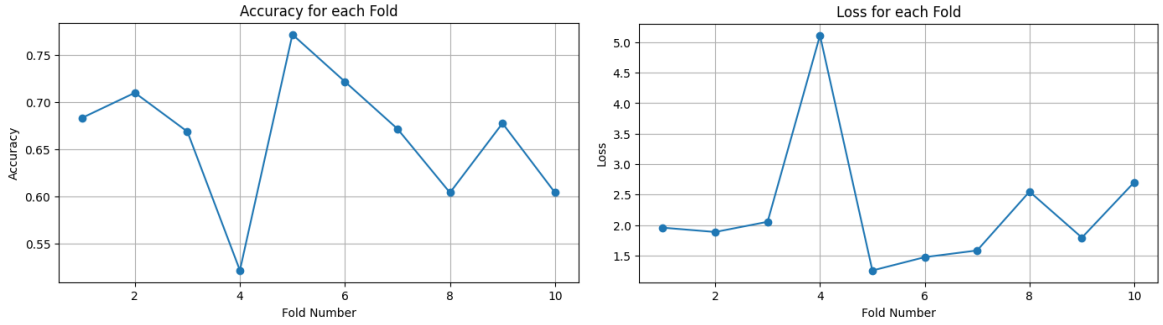
$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (5)$$

The model validation used 10-fold cross-validation to validate the model. The 3410 images are divided into ten folds. Every fold has 341 images, meaning 3069 images are used as training data and 341 as testing data. The testing used ten epochs. Table 2 shows the result of the model validation. The validation model obtained average accuracy, precision, recall, and f1-score of 66.33%, 73.4%, 66.2%, and 66%, respectively.

Table 2. Result of 10 Folds Cross Validation

Fold	Accuracy	Precision	Recall	F1-Score
1	68.32%	74%	68%	68%
2	70.96%	77%	71%	71%
3	66.86%	79%	67%	68%
4	52.19%	54%	52%	49%
5	77.12%	83%	77%	77%
6	72.14%	75%	72%	71%
7	67.15%	71%	67%	66%
8	60.41%	76%	60%	62%
9	67.74%	74%	68%	68%
10	60.41%	70%	60%	60%
Average	**66.33%**	**73.4%**	**66.2%**	**66%**

The accuracy and loss will be visualized after all the folds have been iterated. Figure 7 (a) shows the accuracy graph of each fold. Meanwhile, Figure 7 (b) shows the loss graph of each fold. Based on Figure 7 (a), the best accuracy was obtained in the fifth and worst in the fourth fold. The highest loss was obtained in the fourth fold and the lowest in the fifth fold.



(a) (b)

Figure 7. Visualization of Accuracy and Loss Every Fold

3.2. Increment II

The second increment is used to develop the application. It is divided into four steps: analysis, design, implementation, and testing.

3.2.1. Analysis

The analysis stage at Increment II gathers the requirements to develop the application. The application will implement the model developed in the previous increment. The application must have features that support predicting case data against the model in Increment I. The features that are implemented are the search image button to search for images using File Explorer from Windows and display the selected image and the prediction button to predict the image that has been selected and display the prediction on the second display. The model and application were developed using Python 3.8 and Library Tkinter, PIL, Tensorflow, Keras, and OpenCV.

3.2.2. Design

The next step is design. This stage is used to develop the design of the application. The developed design must meet the needs of the Analysis stage. Figure 8 shows the flow to develop the user interface. It started with an analysis of the application and then developed the UI mockup. After that, implement the mockup into code to be an application user interface.

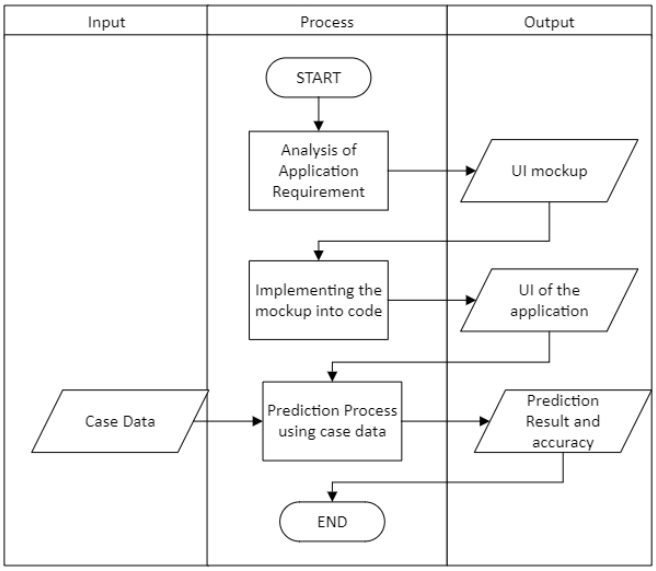


Figure 8. The Flow of Increment II

Figure 9. Mockup the application

Figure 9 is the mockup of the application. It will have two displays. The left display will show the image that will be predicted, and the right one will show the prediction result. Between the two displays, there will be two buttons. They are the “search the image” and the “prediction” buttons.

3.2.3. Implementation

The implementation phase will focus on creating the application. All activities in Incremental I will be applied at this stage. The application development used the Tkinter library to build the application and the PIL library to process images so the Tk widget could display them. The model from Incremental I will be used to predict the input image. Figure 10 is the user interface of the application when it started.

Figure 10. User Interface of The Application

Figure 11 shows the interface when the user wants to insert the image that wants to be predicted or press the "cari gambar" button. It will pop up a new window to search the image.

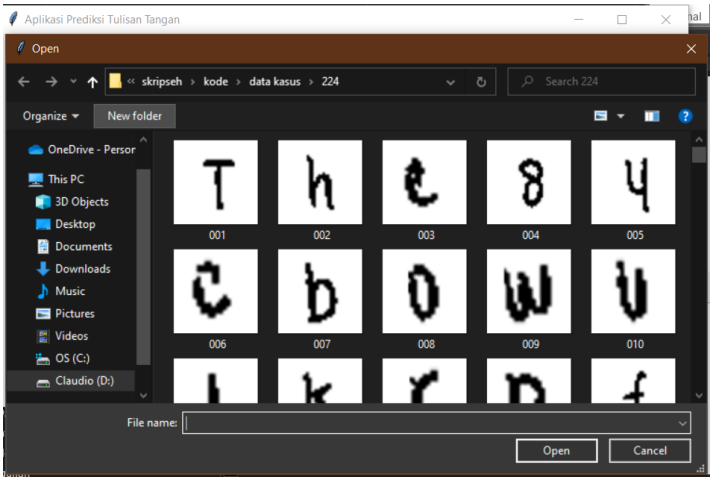


Figure 11. User interface when choosing "Cari Gambar" Button

The file that can be opened must be in format PNG, JPG, or JPEG. When the user tries to choose an image in other formats, the application will notify the user in a pop-up window that the image that is predicted is an invalid format. Figure 12 shows the notification when the image is in an invalid format.

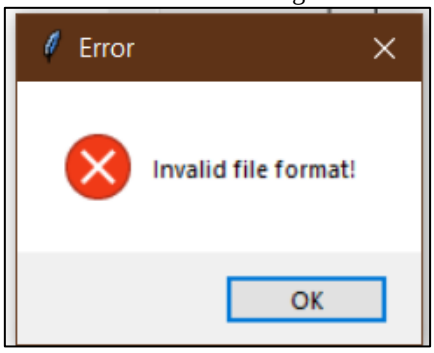


Figure 12. User Interface when choosing an invalid format

Figure 13 shows the interface when the user chooses the right image format. The image will be shown on the right side of the window.

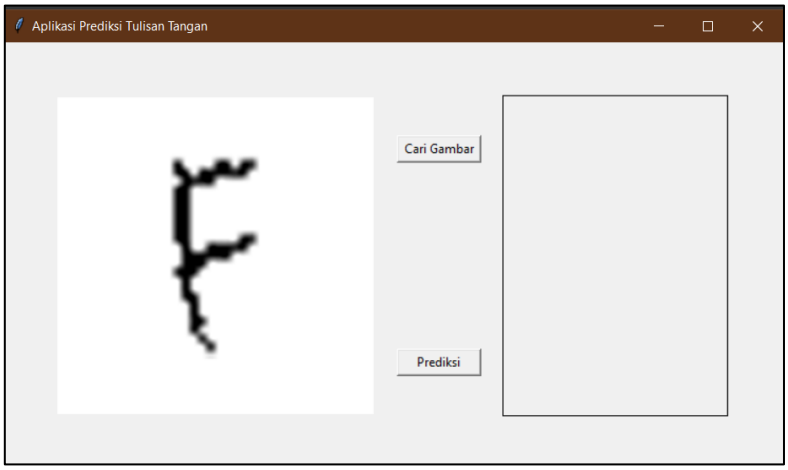


Figure 13. User Interface when choosing a valid format

After the user successfully selects an image that matches the format, the user can press the "Prediction" button to predict the letters according to the selected image. The prediction results will appear on the right side of the window, displaying the top three predictions. Figure 14 shows the interface when the application successfully predicted the image.

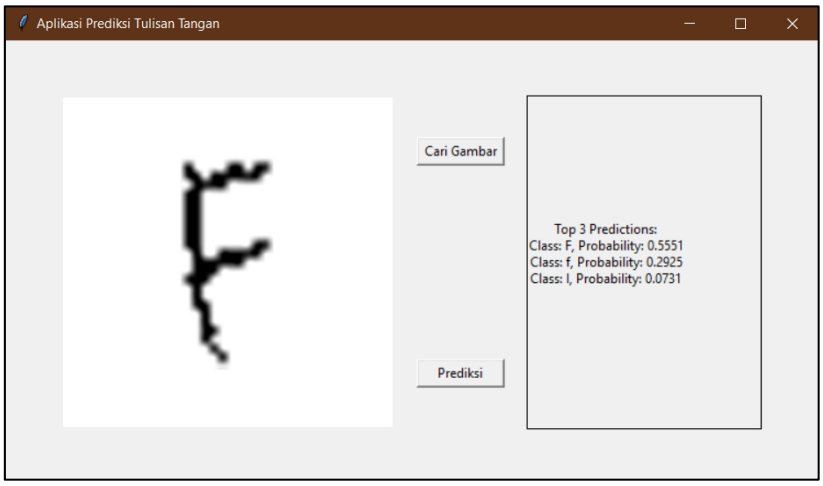


Figure 14. User Interface when successfully predicting the image

3.2.4. Test

Table 4. Result of Black Box Testing

Function that tested	Expected Result	Success/ Failed
"Cari Gambar" Button	The application can load image types PNG, JPG, JPEG	Success
	The application can not load image types other than PNG, JPG, JPEG	Success
	The application can show the image in the main window	Success
"Prediksi" Button	The application can process the image	Success
	The application can show three top predictions and accuracy	Success

Application testing used black box testing. Testing is carried out on all features in the UI to determine the success of each existing function. Black box testing is functional testing of an application based only on application specification information [33]. Table 4 shows the result of the black box testing on application specification. Based on the results, all the functions that were tested are successfully running well.

The case test used 16 handwritten letters from 16 people. The case test data consists of 687 letters with a size of 224x224 pixels. Case test data choose 10 letters randomly to test the accuracy of the model already made. Table 5 shows the result of the test case.

Table 5. Case Test Result

Image	First Probability	Second Probability	Third Probability

	F 55.51%	f 29.25%	l 7.31%
	h 99.82%	A 0.08%	n 0.06%
	N 87.22%	M 12.49%	n 0.07%
	E 82.09%	4 8.12%	9 2.93%
	H 89.15%	M 7.68%	h 2.10%
	A 56.21%	H 22.97%	4 13.85%
	T 58.86%	7 22.42%	J 18.36%
	m 75.56%	M 24.43%	N 0.00%
	1 47.96%	I 40.98%	l 7.98%
	H 44.85	A 17.95%	w 6.05%
	t 56.44%	h 40.45%	H 1.09%
Total	**5**	**1**	**2**

In testing the application prediction using ten case test data, there were five successful attempts at predicting the correct result at the first probability, one attempt at successfully predicting the correct result at the second probability, 2 successful attempts at predicting the correct result at the second probability, and 2 attempts were not classified correctly. The accuracy that the letter successfully predicted as the first prediction in the application is 50%.

4. CONCLUSION

Based on the study conducted, it can be concluded that the application for transforming handwriting into digital was successfully carried out. The model using CNN with ResNet50 architecture obtained accuracy, precision, recall, and f1 scores of 66.33%, 73.4%, 66.2%, and 66%, respectively. Based on the result of black box testing, all the application functionalities went as expected. The case test uses ten data of handwriting images; five can be predicted correctly, or the accuracy is 50%. However, the model and the application still have room for improvement. In the learning model development, need to add more training datasets to get a better model and separate capital letters, lower letters, and numbers so that the training becomes faster and obtains higher accuracy. Besides that, CNN can be improved with different architecture and more layers. For the future work, the application can be developed into mobile application and can recognize all the handwriting characters in the picture.

REFERENCES

[1] A. Prasojo, A. Hidayatno, and R. R. Isnanto, "Pengenalan Karakter Alfabet Menggunakan Jaringan Saraf Tiruan," Universitas Diponegoro, 2011.

[2] N. Majid and E. H. Barney Smith, "Performance comparison of scanner and camera-acquired data for Bangla offline handwriting recognition," *2019 International Conference on Document Analysis and Recognition Workshops, ICDARW 2019*, vol. 2019-January, pp. 31–36, 2019, doi: 10.1109/ICDARW.2019.30061.

[3] A. Korichi, S. Slatnia, O. Aiadi, N. Tagougui, and M. Kherallah, "Arabic handwriting recognition: Between handcrafted methods and deep learning techniques," *Proceedings - 2020 21st International Arab Conference on Information Technology, ACIT 2020*, Nov. 2020, doi: 10.1109/ACIT50332.2020.9300121.

[4] M. Kamal, F. Shaiara, C. M. Abdullah, S. Ahmed, T. Ahmed, and M. H. Kabir, "Huruf: An Application for Arabic Handwritten Character Recognition Using Deep Learning," in *Proceedings of 2022 25th International Conference on Computer and Information Technology, ICCIT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1131–1136. doi: 10.1109/ICCIT57492.2022.10054769.

[5] C. Srinilta and S. Chatpoch, "Multi-task learning and thai handwritten text recognition," *Proceedings - 2020 6th International Conference on Engineering, Applied Sciences and Technology, ICEAST 2020*, Jul. 2020, doi: 10.1109/ICEAST50382.2020.9165315.

[6] N. Nimsuk, N. Thumpaiboon, and W. Phuangsri, "Offline Handwriting Recognition of Thai Characters Using Multiple Deep Neural Networks," *2023 3rd International Symposium on Computer Technology and Information Science, ISCTIS 2023*, pp. 780–785, 2023, doi: 10.1109/ISCTIS58954.2023.10213205.

[7] T. T. Zin, M. Z. Pwint, and S. Thant, "A mobile application for offline handwritten character recognition," in *2020 IEEE 9th Global Conference on Consumer Electronics, GCCE 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 10–11. doi: 10.1109/GCCE50665.2020.9291735.

[8] H. Mule, N. Kadam, and D. Naik, "Handwritten Text Recognition from an Image with Android Application," in *Proceedings of IEEE 2022 4th International Conference on Advances in Electronics, Computers and Communications, ICAECC 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICAECC54045.2022.9716714.

[9] S. Namiki, K. Yokoyama, S. Yachida, T. Shibata, H. Miyano, and M. Ishikawa, "Online Object Recognition Using CNN-based Algorithm on High-speed Camera Imaging: Framework for fast and robust high-speed camera object recognition based on population data cleansing and data ensemble," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, Jan. 2021, pp. 2025–2032. doi: 10.1109/ICPR48806.2021.9413042.

[10] P. Roy, S. Ghosh, and U. Pal, "A CNN Based Framework for Unistroke Numeral Recognition in Air-Writing," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, Aug. 2018, pp. 404–409. doi: 10.1109/ICFHR-2018.2018.00077.

[11] M. Mukhopadhyay, A. Dey, R. N. Shaw, and A. Ghosh, "Facial emotion recognition based on Textural pattern and Convolutional Neural Network," in *2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON)*, IEEE, Sep. 2021, pp. 1–6. doi: 10.1109/GUCON50781.2021.9573860.

[12] P. V Bhagyasree, A. James, and C. Saravanan, "A Proposed Framework for Recognition of Handwritten Cursive English Characters using DAG-CNN," in *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, IEEE, Apr. 2019, pp. 1–4. doi: 10.1109/ICIICT1.2019.8741412.

[13] J. Butdee, W. Kongprawechnon, H. Nakahara, N. Chayopitak, C. Kingkan, and R. Pupadubsin, "Pattern Recognition of Partial Discharge Faults Using Convolutional Neural Network (CNN)," in *2023 8th International Conference on Control and Robotics Engineering (ICCRE)*, IEEE, Apr. 2023, pp. 61–66. doi: 10.1109/ICCRE57112.2023.10155616.

[14] X. Wan, H. Song, L. Luo, Z. Li, G. Sheng, and X. Jiang, "Pattern Recognition of Partial Discharge Image Based on One-dimensional Convolutional Neural Network," in *2018 Condition Monitoring and Diagnosis (CMD)*, IEEE, Sep. 2018, pp. 1–4. doi: 10.1109/CMD.2018.8535761.

[15] D. S. Wita and D. Y. Liliana, "Klasifikasi Identitas Dengan Citra Telapak Tangan Menggunakan Convolutional Neural Network (CNN)," *Jurnal Rekayasa Teknologi Informasi (JURTI)*, vol. 6, no. 1, p. 1, Jul. 2022, doi: 10.30872/jurti.v6i1.7100.

[16] N. Kasim and G. S. Nugraha, "Pengenalan Pola Tulisan Tangan Aksara Arab Menggunakan Metode Convolution Neural Network," *Jurnal Teknologi Informasi, Komputer, dan Aplikasinya (JTIKA)*, vol. 3, no. 1, pp. 85–95, Apr. 2021, doi: 10.29303/jtika.v3i1.136.

[17] D. Fitriati, "PERBANDINGAN KINERJA CNN LeNet 5 DAN EXTREME LEARNING MACHINE PADA PENGENALAN CITRA TULISAN TANGAN ANGKA," *Jurnal Teknologi Terpadu*, vol. 2, no. 1, Jul. 2016, doi: 10.54914/jtt.v2i1.45.

[18] C. Umam and L. Handoko, "CONVOLUTIONAL NEURAL NETWORK (CNN) UNTUK IDENTIFKASI KARAKTER HIRAGANA," *PROSIDING SEMINAR NASIONAL LPPM UMP*, vol. 2, no. 0, pp. 527–533, 2021, [Online]. Available: http://semnaslppm.ump.ac.id/index.php/semnaslppm/article/view/199

[19] "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more… | by Siddharth Das | Analytics Vidhya | Medium." Accessed: Nov. 08, 2023. [Online]. Available:

https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5
[20] S. Li, J. Jiao, Y. Han, and T. Weissman, "Demystifying ResNet." 2017.
[21] R. Jayakanthan, A. H. Kumar, N. Sankarram, B. S. Charulatha, and A. Ramesh, "Handwritten tamil character recognition using ResNet," *International Journal of Research in Engineering, Science and Management*, vol. 3, no. 3, pp. 133–137, 2020.
[22] V. Atliha and D. Sesok, "Comparison of VGG and ResNet used as Encoders for Image Captioning," in *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, IEEE, Apr. 2020, pp. 1–4. doi: 10.1109/eStream50540.2020.9108880.
[23] A. Rachman, Andreansyah, and Rahmi, "Implementation of Incremental Models on Development of Web-Based Loan Cooperative Applications," *International Journal of Education, Science, Technology, and Engineering*, vol. 3, no. 1, pp. 26–34, May 2020, doi: 10.36079/lamintang.ijeste-0301.105.
[24] "English Handwritten Characters Dataset | Kaggle." Accessed: Oct. 06, 2023. [Online]. Available: https://www.kaggle.com/discussions/general/221624
[25] Y. Jung, "Multiple predicting *K* -fold cross-validation for model selection," *J Nonparametr Stat*, vol. 30, no. 1, pp. 197–215, Jan. 2018, doi: 10.1080/10485252.2017.1404598.
[26] "Incremental Model in SDLC: Use, Advantage & Disadvantage." Accessed: Nov. 02, 2021. [Online]. Available: https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html
[27] "Incremental Model in SDLC: Use, Advantage & Disadvantage." Accessed: Oct. 08, 2023. [Online]. Available: https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html
[28] V. Atliha and D. Sesok, "Comparison of VGG and ResNet used as Encoders for Image Captioning," in *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, IEEE, Apr. 2020, pp. 1–4. doi: 10.1109/eStream50540.2020.9108880.
[29] E. S. Marquez, J. S. Hare, and M. Niranjan, "Deep Cascade Learning," *IEEE Trans Neural Netw Learn Syst*, vol. 29, no. 11, pp. 5475–5485, Nov. 2018, doi: 10.1109/TNNLS.2018.2805098.
[30] "Vanishing Gradient Problem | What is Vanishing Gradient Problem?" Accessed: Oct. 10, 2023. [Online]. Available: https://www.mygreatlearning.com/blog/the-vanishing-gradient-problem/
[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, Dec. 2015, doi: 10.1109/CVPR.2016.90.
[32] O. Caelen, "A Bayesian interpretation of the confusion matrix," *Ann Math Artif Intell*, vol. 81, no. 3–4, pp. 429–450, Dec. 2017, doi: 10.1007/S10472-017-9564-8/METRICS.
[33] S. Nidhra, "Black Box and White Box Testing Techniques - A Literature Review," *International Journal of Embedded Systems and Applications*, vol. 2, no. 2, pp. 29–50, Jun. 2012, doi: 10.5121/ijesa.2012.2204.

BIOGRAPHIES OF AUTHORS

	Claudio Josulin is an alumnus of a Bachelor's Degree in Informatics majoring in Mobile Computing at Institut Teknologi dan Bisnis Kalbis (Kalbis Institute) Jakarta who graduated in July 2023.

	Yulia Ery Kurniawati is an Informatics lecturer at Institut Teknologi dan Bisnis Kalbis (Kalbis Institute) Jakarta. She obtained her Bachelor's Degree in Informatics from Universitas Sebelas Maret Surakarta and her Master's Degree in Information Technology from Universitas Gadjah Mada Yogyakarta. Her research focuses on artificial intelligence in machine learning and class imbalance learning.