



The Implementation of Artificial Neural Network (ANN) on Offline Cursive Handwriting Image Recognition

Fitrianingsih^{a,1,*}; Diana Tri Susetianingtias^{a,2}; Dody Pernadi^{a,3}; Eka Patriya^{a,4}; Rini Arianty^{a,5}

^a Gunadarma University, Margonda Raya Street 100, Pondok Cina Depok, Indonesia

ⁿ Gunadarma University, Margonda Raya Street 100, Pondok Cina Depok, Indonesia

¹ fitrianingsih@staff.gunadarma.ac.id; ² diants@staff.gunadarma.ac.id; ³ dody_ernadi@staff.gunadarma.ac.id;

⁴ ekapatriya@staff.gunadarma.ac.id; ⁵ rinia@staff.gunadarma.ac.id;

* Diana Tri Susetianingtias

Article history: Received February 14, 2022; Revised March 3, 2022; Accepted March 22, 2022; Available online March 22, 2021

Abstract

Identifying a writing is an easy thing to do for human, but this does not apply to computers, in particular if it is handwriting. Handwriting recognition, especially cursive handwriting is a research in the area of image processing and pattern matching that is challenging to complete, following the different characteristics of each person's cursive handwriting style. In this study, the use of the ANN model will be implemented in performing offline handwriting image recognition. The cursive handwriting image that has been obtained is then preprocessed and segmented using bounding box rectangle and contour techniques. Evaluation of system performance using global performance metrics in this study resulted in a percentage of 93% where the bounding box and contour succeeded in determining the segmentation point correctly, so that the ANN model worked optimally.

Keywords: ANN; Global Performance Metric; Contour; Segmentation; Writing.

Introduction

Handwriting recognition especially the cursive type is research in the area of image processing and pattern matching that is very difficult to conduct because the characteristics of each person's cursive handwriting [1] could be various such as [2]: alphabet writing style, letter size, direction writing, thickness, etc. [3]. This [4] becomes an obstacle in the current cursive handwriting recognition system [5].

One method that can be used to recognize cursive handwriting is Artificial Neural Network (ANN) [6] [7]. ANN is a neural network [8] which consists of a large number of information processing elements (neurons) [9] [10] which are interconnected and work together to solve a particular problem, including the problem of handwriting image recognition [11] [2] [13]. Several researches on to offline cursive handwriting recognition have been previously carried out. Research [14] applied ANN to identify offline cursive handwriting using the gradient descent back propagation method with an accuracy of 97%. On the other hand, study [15] applied ANN to identify offline Arabic cursive handwriting using the feed forward neural network method at the classification stage with an accuracy of 83% for all characters and 96% for some characters. Another research [16] applied ANN to identify cursive handwriting and offline non-continuous handwriting using the Support Vector Machine (SVM) algorithm with an accuracy of 97.2%. Research [17] applied ANN and Hidden Markov Model to identify Malayalam cursive handwriting (South Indian language) with 93.4% accuracy. Research [18] applied ANN and Genetic Algorithm to identify offline Arabic cursive handwriting with 98% accuracy. Research [2] applied ANN and SVM to identify offline cursive handwriting with 98% accuracy.

This study applies the ANN method in offline handwriting recognition. The cursive handwriting image used was obtained from the respondent who wrote cursively. Data acquisition was then carried out and produced a cursive handwriting image in .JPG format. Segmentation was done using bounding box rectangles and contours. The modeling process was carried out using TensorFlow and hard. The recognition model used was the ANN architecture. The results of the study are expected to be able to recognize offline cursive handwriting.

Methods

This research was carried out in several stages to perform offline cursive handwriting recognition using the ANN model as can be seen in Figure 1. The first stage carried out was to input data in the form of 92 cursive handwriting images that had gone through the scanning process and saved in .JPG format. The next stage was preprocessing the cursive handwritten image, which consists of several processes such as grayscale, binaryzation, edge detection, dilation morphology operations, thinning and deskew. After the preprocessing stage, the segmentation stage using bounding box and contour techniques was carried out. The last stage in was introduction. At this stage, trials were carried out on the system which was expected to be able to recognize cursive handwriting. The stages in this research can be seen in Figure 1.

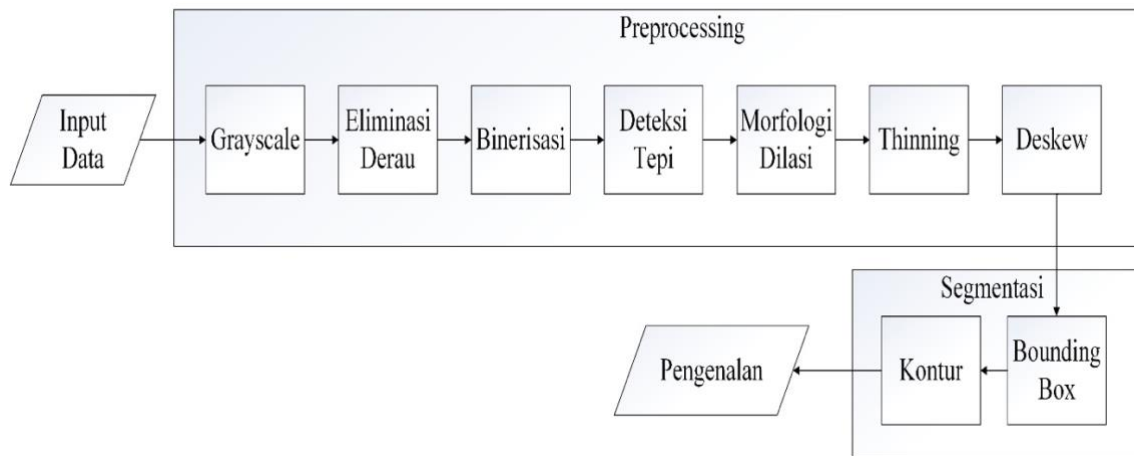


Figure 1. Research Methods (Stages)

A. Dataset of handwriting images

The dataset used in this study was cursive handwriting images obtained from respondents who wrote cursive handwriting. Then, the data acquisition process was carried out on cursive handwriting images and stored in .JPG format with a total of 300 images. The cursive handwriting images consist of 1-word handwriting images and 2-word handwritten image. An example of the dataset used can be seen in Figure 2.

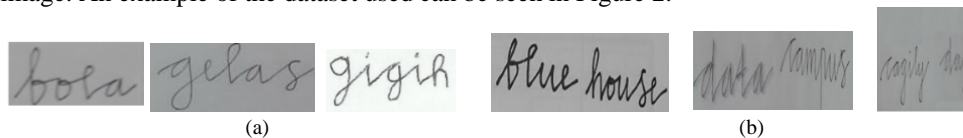


Figure 2 Example of Images used. (a). 1-word image, (b). 2-word image

B. Preprocessing of cursive handwriting images

The initial stage was preprocessing which was conducted in several steps as follows:

1. Converting an RGB image to grayscale as can be seen in Figure 3. This study applied the OpenCV library to perform the grayscale process on a cursive handwriting image. Grayscale implementation with the `cvtColor()` function [19]. The following pseudocode used OpenCV to process grayscale images.

```

img = cv.imread(dir_path + file)
gray_img = cv.cvtColor(img,
cv.COLOR_BGR2GRAY)
  
```

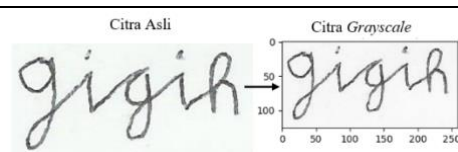


Figure 3 Grayscale Image Process

- Conducting binary process on grayscale image so that the image has only 0 or 1 value. This process used library OpenCV which is the threshold function [20] as shown in the following pseudocode snippet:

```
def _binarize(self, img):
    _, threshhold_img = cv.threshold(img,0,255,
cv.THRESH_OTSU)
    return threshhold_img
```

Declaration of a function with the name of `_binarize` with parameter of `self` and `img`, then declare a variable of `threshhold_img` which served to store the value of the function `cv.threshold(img,0,255, cv.THRESH_OTSU)` consisting of 4 parameters [21]. The first parameter was a cursive handwriting image to be processed; the second parameter was the value of the threshold, which is 0; the third parameter was the maximum value of the threshold, which is 255 [22]; and the fourth parameter was the type of threshold used, namely `thresh_otsu` [23]. Then, function call was conducted as follow:

```
binarized_img = self._binarize(denoised_img)
```

The binarization process of cursive handwriting image can be seen in Figure 4

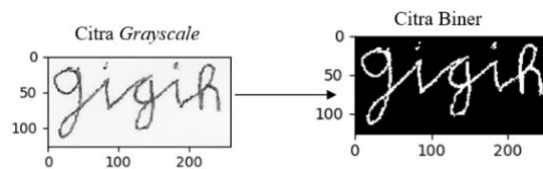


Figure. 4 Image Binarization Process

- Performing binary image edge detection to identify the boundary line of an object contained in a cursive handwriting image using the `canny()` and `sobel()` operators [24] by the following pseudocode:

```
def _slant_corrector(self, img):
    edges = cv.Canny(img, 100, 200)
    img_sobelx = cv.Sobel(edges, cv.CV_8U, 1, 0,
ksize=0)
    img_sobely = cv.Sobel(edges, cv.CV_8U, 0, 1,
ksize=0)
    img_sobel = img_sobelx + img_sobely
    return img_sobel
```

Edge detection was done by declaring the `slant_corrector` function with `self` and `img` parameters, by declaring the `edges` variable which was used to store the value of the `cv.Canny(img, 100, 200)` function which consisted of 3 parameters. The first parameter was the cursive handwriting image to be processed, the second one was the value of threshold 1 which was 100, and the third parameter was the value of threshold 2 which was 200. Edges that have been processed using the `canny()` function [25] were then processed using `sobel` by declaring `img_sobelx = cv.Sobel(edges, cv.CV_8U, 1, 0, ksize=0)` and `img_sobely = cv.Sobel(edges, cv.CV_8U, 0, 1, ksize=0)` to calculate edges both horizontally and vertically. The `sobel x` operator parameter consisted of 5 parameters [26]: the first parameter was a cursive handwriting image to be processed, the second parameter was unsigned 8 bits, the third parameter was x-order with the value of 1, the fourth parameter was y-order with the value of 0, and the last parameter was the size of the sobel kernel which was 0. The parameters used in the `sobel y` operator in this study consisted of 5 parameters [27]: the first parameter was a cursive handwriting image to be processed, the second parameter was unsigned 8 bits, the third parameter was x-order with a value of 0, the fourth parameter is y-order which was 1, and the last parameter was the size of the sobel kernel which was 0. After declaring `img_sobelx` and `img_sobely`, the variable `img_sobel = img_sobelx + img_sobely` was then declared in order to compute the value of edge detection horizontally and vertically. The edge detection process in the cursive handwriting image can be seen in Figure 5.



Figure. 5 Image Edge Detection Process using Operator Canny and Sobel

4. Performing a dilation morphology process in order to enhance the cursive handwriting image in the thinning process [28] and at the segmentation stage, using the function in OpenCV dilate().

```
def _dilate_image(self, img):
    dilated_img = cv.dilate(img, self.kernel, iterations = 1)
    return dilated_img
```

The `dilated_img` variable was used to store the value of the `cv.dilate(img, self.kernel, iterations = 1)` function which consisted of 3 parameters. The first parameter was a cursive handwriting image to be processed; the second parameter was the value of the kernel used; and the third parameter was the iteration value of the dilation to be performed. This study applied the dilation iteration value of 1. The morphological process of dilation [29] on the cursive handwriting image can be seen in Figure 6.

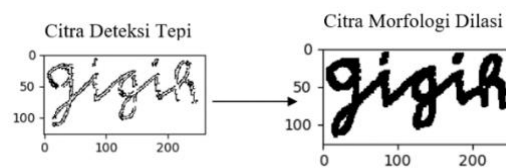


Figure. 6 Image Dilation Morphology Operation Process

5. Perform a thinning process to obtain a one-pixel-sized image that would be used in the segmentation process. The thinning process was carried out with the function in OpenCV, namely `ximgproc.thinning()` as can be seen in the following code snippet:

```
def _thining_image(self, img):
    thinning_image = cv.ximgproc.thinning(img)
    return thinning_image
```

The `thinning_image` variable was used to store the value of the `cv.ximgproc.thinning(img)` function which consisted of 1 parameter, the cursive handwriting image to be processed. The thinning process on the cursive handwriting image can be seen in Figure 7.

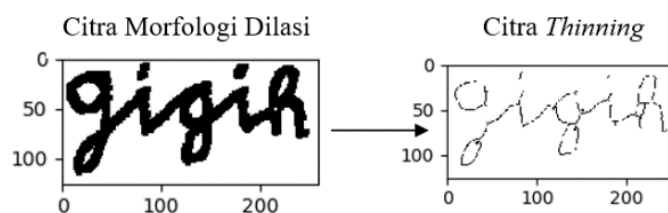


Figure. 7 Image Thinning Process

C. Segmentation of cursive handwriting image

The segmentation process was carried out to determine the right cut point by using the image from the previous thinning process. Images from segmentation process were in the form of image pieces of letters that will be used in the image recognition process of cursive handwriting. Segmentation was performed using the following techniques:

1. Bounding box is an imaginary box around the handwriting image object so that the object has an area that is not wide. The implementation of the bounding box technique is carried out using the `cv.boundingRect()` function as can be seen in the following pseudocode:

```
boundingBoxes = [cv.boundingRect(c) for c in contours]
```

This function will loop over the contour with a contour as the parameter of the bounding box.

2. The next process after finding the bounding box is to determine the contour using `cv.findContours()` through the following pseudocode:

```

contours, _ = cv.findContours(
    final_thr,
    cv.RETR_EXTERNAL,
    cv.CHAIN_APPROX_SIMPLE
)
boundingBoxes = [cv.boundingRect(c) for c in contours]
(contours, boundingBoxes) = zip(
    *sorted(
        zip(contours, boundingBoxes), key=lambda b:
b[1][0], reverse=False))

for cnt in contours:
    area = cv.contourArea(cnt)
    if area > 100000:
        print('contours-----', cnt)
        print('area-----', area)
        print('Area= ',area)
        x,y,w,h = cv.boundingRect(cnt)
        print(x,y,w,h)
        letterBgr = txt_line[0:txt_line.shape[1],x:x+w]
        wordImgList.append(letterBgr)
        words_result_path =
"./result/words/{ }.jpg".format(self.words_count)
        cv.imwrite(words_result_path,letterBgr)
        self.words_count += 1
    cl=cl+1

```

The pseudocode was used to find a contour in a cursive handwriting image with three parameters. The first parameter was a cursive handwriting image, the second parameter was the type of contour capture, namely `cv.RETR_EXTERNAL`, and the last parameter was used to determine the point of the contour. After finding the contour, the next step was to declare a bounding box to wrap the contour found in the cursive handwriting image. Then, sorting was done on the contour and the bounding box. The next process was to iterate over the contour and to find the area of the contour in the cursive handwriting image provided that if the area is more than 100000 then a bounding box is given to wrap the contour. After that, the cursive handwriting image was stored in the words folder. The segmentation algorithm which is to determine the exact point of intersection of the letters from the cursive handwriting image was carried out by the following algorithm:

1. Read the image of the preprocessing process.
2. Determine the bounding box and contour of the preprocessing image.
3. Determine the minimum value of the pixel threshold, the minimum value of the separation threshold, and the minimum value of the round letter threshold.
4. Create functions for histogram, `most_frequent`, and `baseline`
5. Find the `most_frequent` range of contour values from the histogram.
6. Compare the value of the `most_frequent` contour of the histogram obtained. If the value of the `most_frequent` range of the contour of the histogram < the minimum pixel threshold, it will enter segment 1. But, if the value of segment 1 > of the minimum value of the separation threshold then enter segment 2.
7. Eliminate circles in the segmentation area.
8. After removing the circles, cut the sentence into letters and save the illustration of the segmentation visualization into the `seg_visualize` folder.

The segmentation process of cursive handwriting image can be seen in. Figure 8.

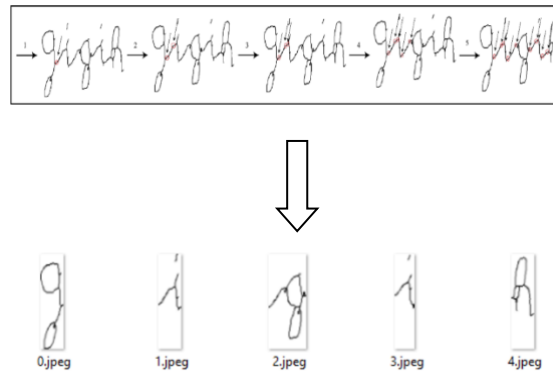


Figure. 8 Image Segmentation Process

D. Cursive handwriting image recognition with ANN

Recognition is the third or the final stage in this study. This was carried out to identify the reading of cursive handwriting image by a computer, by implementing the following pseudocode:

```
from matplotlib import pyplot as plt
import numpy as np
from keras.models import load_model
import cv2
import os
import numpy
from keras import backend as K
```

This study imported matplotlib, numpy, keras[28], OpenCV, and os library with the following pseudocode:

```
K.common.set_image_dim_ordering('th')

model=load_model('./data/modelnewv1.h5')
print(model.summary())

import string
letter_count = dict(zip(string.ascii_lowercase, range(1,27)))
print('Letter_count: ',letter_count.items())

x=[]
res=[]
fname=[]
folder='./result/resized_images/'
dirFiles=os.listdir(folder)
dirFiles = sorted(dirFiles,key=lambda x:
int(os.path.splitext(x)[0]))
```

After importing the library, the model and the resulting letter image were the imported at the segmentation process using the following pseudocode:

```
for filename in dirFiles:
    imt = cv2.imread(os.path.join(folder,filename))
    gray = cv2.cvtColor(imt,cv2.COLOR_BGR2GRAY)
    _, imt = cv2.threshold(gray,0,255,cv2.THRESH_OTSU)
    if imt is not None:
        imt = imt.reshape((-28))
        imt=imt/255
        x.append(imt)
        fname.append(filename)

x=np.array(x);
predictions = model.predict(x)
classes = np.argmax(predictions, axis=1)
```

```

for i in range(len(classes)):
    imt = cv2.imread(os.path.join(folder,dirFiles[i]))
    print([k for k,v in letter_count.items() if v == classes[i]])

```

Next, the cursive handwriting image was converted into an array form, which was then repeated. The final stage was to perform the image recognition of the cursive handwriting.

E. Calculation of Cursive handwriting image recognition accuracy

The performance evaluation of the implemented algorithms consisted of:

1. Letter segmentation accuracy

Segmentation performance in this study was measured using the Performance Metric Segmentation (FMS) [1] parameter which is a comparison of the results of two trial processes between successful segmentation of each letter (Detection Rate Segmentation/DRS) and the sum of the true letter segmentation recognition (Recognition Accuracy Segmentation/ RAS). The total segmentation and the recognition of each letter segmentation results can be seen in the following equation [1]:

$$FMS = \frac{2DRS.RAS}{DRS + RAS} \quad (1)$$

The values of the Detection Rate Segmentation and Recognition Rate Segmentation were calculated using the number of groundtruth segmentations, the number of segmentation results from the program, and the number of true segmentation results from the program. Detection Rate Segmentation is the comparison of the number of segmentation results from the program with the number of groundtruth segmentations, while Recognition Accuracy Segmentation is the comparison of the number of true segmentation results from the program with the number of segmentation results from the results of program trials using equation [1]:

$$DRS = \frac{RS}{GS} \quad (2)$$

$$RAS = \frac{RS}{MS} \quad (3)$$

Where GS is the number of groundtruth segmentation which is the number of true segmentation program results and the number of segmentation program results.

2. Image recognition accuracy

Letter recognition performance in this study was measured using the Recognition Performance Metric parameter (), which is a comparison of the results of the multiplication of the two trial processes between successful each letter recognition (Detection Rate Recognition / DRP) and the sum of true letter recognition (Recognition Accuracy / RA). The total recognition and the results of each letter recognition can be seen in the following equation [1]:

$$FMP = \frac{2DRP.RA}{DRP + RA} \quad (4)$$

The Detection Rate Recognition (DRP) and Recognition Accuracy (RA) values were performed using the parameters of the number of letters recognized by the program, the number of letters of groundtruth, and the number of letters correctly recognized by the program. Detection Rate Recognition is a comparison of the number of letters recognized by the program with the number of groundtruth letters, while Recognition Accuracy is a comparison of the number of letters recognized by the program with the number of letters correctly recognized resulting from the correct program test using equation [1]:

$$DRP = \frac{RP}{GP} \quad (5)$$

$$RA = \frac{RP}{MP} \quad (6)$$

Where GP is the number of groundtruth letters, RP is the number of letters correctly recognized by the program and MP is the number of letters recognized by the program.

Results and Discussion

The experiment was carried out on 300 cursive handwriting images. For example, the results of segmenting the cursive handwriting image into letter pieces and recognition can be seen in Table 1.

Table 1. Images of Segmentation and Recognition Results



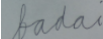

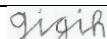



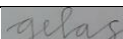

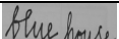
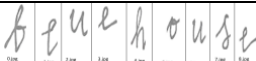
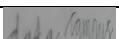
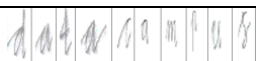




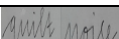



No.	Original Image	Image from Segmentation	Recognition Results
1			bola
2			badai
3			gigih
4			joke
5			gelas
6			blue house
7			data campus
8			tasbih kakek
9			cagily day
10			quite noise






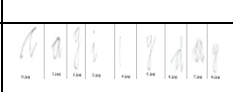
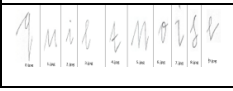
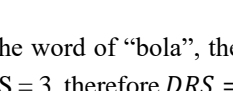
Table 1 shows the original image and the image after the segmentation process on the cursive handwriting image. The original image is a 1-word image, while the image from the segmentation process is an image of splitting 1-word into letters. Table 1 illustrates that the bounding box and contour techniques were successful in the segmentation process and the ANN implementation was successful in recognizing the letter image resulting from the segmentation process.

A. Segmentation Accuracy Results by Global Performance Metrics

Based on the results of the cursive handwriting recognition trials that have been carried out, the performance of the recognition system that has been built can be obtained by calculating the Performance Metric (FM) and Global Performance Metric (SM) values using the parameters of Detection Rate (DR) and Recognition Accuracy (RA) as shown in Table 2.

Table 2. Recognition Accuracy Results

No.	Segmentation Results Image	Recognition Results	GS	RS	MS
1		bola	3	3	3
2		badai	4	4	4

No.	Segmentation Results Image	Recognition Results	GS	RS	MS
3		gigih	4	4	4
4		joke	3	3	3
5		gelas	4	4	4
6		blue house	4	4	4
7		data campus	4	4	4
8		tasbih kakek	4	4	4
9		cagily day	3	3	3
10		quite noise	4	4	4

As example, in the image of the word of “bola”, the results of the segmentation test was obtained by the following figures: GS = 3, RS = 3, and MS = 3, therefore $DRS = \frac{RS}{GS} = \frac{3}{3} = 1$, and $RAS = \frac{RS}{MS} = \frac{3}{3} = 1$. The Performance Metric Segmentation (FMS) value of the entire “bola” handwriting image is given as $FMS = \frac{2DRS.RAS}{DRS+RAS} = \frac{2.1.1}{1+1} = 1 \times 100\% = 100\%$.

In the image that contains the word of “badai”, the results of the segmentation test was obtained by the following figures: GS = 4, RS = 4, and MS = 4, therefore $DRS = \frac{RS}{GS} = \frac{4}{4} = 1$, and $RAS = \frac{RS}{MS} = \frac{4}{4} = 1$. The Performance Metric Segmentation (FMS) value of the entire “badai” handwriting image is given as $FMS = \frac{2DRS.RAS}{DRS+RAS} = \frac{2.1.1}{1+1} = 1 \times 100\% = 100\%$

In the image that contains the word of “gigih”, the results of the segmentation test was obtained by the following figures: GS = 4, RS = 4, and MS = 4, therefore $DRS = \frac{RS}{GS} = \frac{4}{4} = 1$, and $RAS = \frac{RS}{MS} = \frac{4}{4} = 1$. The Performance Metric Segmentation (FMS) value of the entire “gigih” handwriting image is given as $FMS = \frac{2DRS.RAS}{DRS+RAS} = \frac{2.1.1}{1+1} = 1 \times 100\% = 100\%$.

B. Letter Recognition Accuracy Results

Table 3 shows the results of system performance based on the parameters Detection Rate (DR), Recognition Accuracy (RA), Performance Metric (FM) both in the segmentation and letter recognition processes and the overall system performance (Global Performance -SM) on 1-word handwriting images.

Table 3. Global Performance of Cursive Handwriting

Parameter	Letter Segmentation	Letter Recognition
Detection Rate	0.951035197	0.90678
Recognition Accuracy	0.950543478	0.96159
Performance Metric	0.968787848	0.84859177
Global Performance	93%	

Conclusion

The value of Global Performance measured using the parameters Detection Rate (DR), Recognition Accuracy (RA) and Performance Metric shows a segmentation accuracy value of 96%. The implementation of bounding box and contour techniques was successful in determining letter segmentation points so that the use of the ANN model in offline cursive handwriting recognition can work well, where the Global Performance value shows the recognition percentage of 93%. Segmentation errors occur in several cases due to the lack of training data on the letter r so that the implemented algorithm divides 1 letter into several parts.

Further development can be done to improve segmentation and recognition of cursive handwriting by adding training data for each possible cursive letter from various author characteristics, using standardized datasets so that the preprocessing stage is better, and adding other algorithms at the segmentation stage to resulting in better segmentation accuracy so that the recognition process becomes better.

Referensi

- [1] Fitrianiingsih, S. Madenda, Ernastuti, S. Widodo, Rodiah, "Cursive handwriting segmentation using ideal distance approach," *International Journal of Electrical and Computer Engineering*, Vol7, No.5, pp. 2863-2872, October 2017.
- [2] Fitrianiingsih, S. Widodo, S. Madenda, Rodiah, "Slant correction and detection for offline cursive handwriting using 2d affine transform," *International Journal of Engineering Research & Technology (IJERT, ISSN: 2278-0181, International Journal of Engineering Research & Technolog*, Vol 5, Issue 08, pp. 568-572, 2016.
- [3] Monika, M. Ingole, and K. Tighare, "Handwritten character recognition using neural network, " *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, Vol 7 Issue 4, pp. 203-207, July 2021.
- [4] T.S. Gunawan, A.F.R.M. Noor, and M. Kartiwi, "Development of english handwritten recognition using deep neural network," *Indonesian Journal of Electrical Engineering and Computer Science*, vo. 10, no 2, pp. 562-568, 2018.
- [5] A. Boukharouba, and A. Bennia, "Novel feature extraction technique for the recognition of handwritten digits," *Applied Computing and Informatics*, Vol 13(1), pp. 19- 26, 2017.
- [6] R.G. Khalkar, A.S. Dikhit, A. Goel, and M. Gupta, "Handwritten text recognition using deep learning (CNN & RNN)," *International Advanced Research Journal in Science, Engineering and Technology*, Vol. 8, Issue 6, pp. 870-881, June 2021.
- [7] U. Dwivedi, P. Rajput, M.K. Sharma, and G. Noida, "Cursive handwriting recognition system using feature extraction and artificial neural network," *International Research Journal of Engineering and Technology*, vol 4(03), pp. 2202-2206, 2017.
- [8] V. Kurkov, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, "Artificial neural networks and machine learning," *ICANN 2018: 27th International Conference on Artificial Neural Networks*, Rhodes, Greece, October 4-7, Proceedings. Basingstoke, England: Springer, 2018.
- [9] S. Aqab, and M.U. Tariq, "Handwriting recognition using artificial intelligence neural network and image processing," *International Journal of Advanced Computer Science and Applications*, Vol. 11, No. 7, 2020.
- [10] R.R. Nair, N. Sankaran, B.U. Kota, S. Tulyakov, S. Setlur, and V. Govindaraju, "Knowledge transfer using Neural network based approach for handwritten text recognition," In 2018 13th IAPR International Workshop on Document Analysis Systems (DAS) (pp. 441- 446). IEEE, 2018.
- [11] M. P. Varma, S. J. Hore, C. Uday, S. O. Reddy and V. J. Pillai, "Optimized handwritten character recognition using artificial neural network," *International Journal of Scientific & Technology Research*, ISSN 2277-8616, Vol 9, Issue 01, January 2020.
- [12] M. Abdulllah, A. Agal, M. Alharthi, and M. Alrashidi, "Retracted: Arabic handwriting recognition using neural network classifier," *Journal of Fundamental and Applied Sciences*, vol 10(4S), pp. 265-270, 2018.
- [13] N.A. Hamid, and N.A.A. Sjarif, "Handwritten Recognition Using SVM, KNN and Neural Network", arXiv preprint arXiv:1702.00723, 2017.
- [14] U. Dwivedi, P. Rajput, and M. K. Sharma, "Cursive handwriting recognition system using feature extraction and artificial neural network," *International Research Journal of Engineering and Technology*, vol 04, pp. 2202-2206, 2017.
- [15] A. Mohsin, and M. Sadoon, "Developing an arabic handwritten recognition system by means of artificial neural network," *Journal of Engineering and Applied Sciences*, vol 15(1), pp. 1-3, 2020.
- [16] N. Sharma, P. Agarwal, and U. Pandey, "Offline handwriting recognition using neural networks," *International Journal of Advance Research*, 4(2), 1541-1545, 2018.

-
- [17] N.T. Kishna, and S. Francis, "Intelligent tool for malayalam cursive handwritten character recognition using artificial neural network and hidden markov model," International Conference On Inventive Computing And Informatics, pp. 595–598, 2017.
- [18] K. Nahar, "Offline Arabic hand-writing recognition using artificial neural network with genetics algorithm," International Arab Journal of Information Technology, vol 15(4), pp. 701–707, 2018.
- [19] J. Cai, E. Sun, and Z. Chen, "OCR Service Platform Based on OpenCV", Journal of Physics: Conference Series, 1883 012043, 2021.
- [20] S. Garg, K. Kumar, N. Prabhakar, and A. Ratan, "Optical Character Recognition using Artificial Intelligence", International Journal of Computer Applications, vol 179(31), pp 14-20, 2018.
- [21] S. Gong, Z. Huan, M. Ji, X. Chen, and Y. Bao, "ITLCS based on opencv image processing technology", Journal of Physics: Conference Series, 2143 (2021) 012031, 2021.
- [22] A. Mordvintsev, and Abid K., "OpenCV-Python Tutorials Documentation Release 1," [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/_/downloads/en/stable/pdf/, 2021.
- [23] F. Jalled, and I. Varonkov, "Object Detection using Image Processing," Nov. 2016, [Online]. Available: <http://arxiv.org/abs/1611.07791>, 2016.
- [24] L. H. Gong, C. Tian, W. P. Zou, and N. R. Zhou, "Robust and imperceptible watermarking scheme based on Canny edge detection and SVD in the contourlet domain," Multimedia Tools and Applications, vol. 80, no. 1, pp. 439–461, 2021.
- [25] T.B. Arnold, "KerasR: r interface to the keras deep learning library," The Journal of Open Source Software, 2(14), pp. 296, 2017.
- [26] M. Gandhi, J. Kamdar, and M. Shah, "Preprocessing of non-symmetrical images for edge detection," Augmented Human Research, vol. 5, no. 1, pp. 1–10, 2020.
- [27] B. Watkins and A. van Niekerk, "A comparison of object-based image analysis approaches for field boundary delineation using multi-temporal Sentinel-2 imagery," Computers and Electronics in Agriculture, vol 158, pp. 294–302, 2019.
- [28] B. Iqbal, W. Iqbal, N. Khan, A. Mahmood, and A. Erradi, "Canny edge detection and hough transform for high resolution video streams using hadoop and spark," Cluster Computing, vol. 23, no. 1, pp. 397–408, 2020.
- [29] J. Chaki, and N. Dey, "A beginner's guide to image preprocessing techniques," Boca Raton: CRC Press. ISBN: 978-1-138-33931-6, 2018.
- [30] M. Dawson, "Python programming: for the absolute beginner (3rd ed)," Boston, MA: Course Technology Cengage Learning. ISBN: 978-1-4354-5500-9, 2010.