IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.15, No.3, July 2021, pp. 221~232
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: https://doi.org/10.22146/ijccs.66062

Transliteration of Hiragana and Katakana Handwritten Characters Using CNN-SVM

Nicolaus Euclides Wahyu Nugroho*[1], Agus Harjoko[2]
[1]Master Program of Computer Science, FMIPA UGM, Yogyakarta, Indonesia
[2]Department of Computer Science and Electronics, FMIPA UGM, Yogyakarta, Indonesia
e-mail: *[1]**nicolaus.euclides@mail.ugm.ac.id**, [2]aharjoko@ugm.ac.id

Abstrak

Karakter tulisan tangan hiragana dan katakana sering digunakan dalam menuliskan kata dalam bahasa Jepang. Bahasa Jepang sendiri sering digunakan oleh orang Jepang asli maupun orang yang belajar bahasa Jepang di seluruh dunia. Karakter hiragana dan katakana sendiri susah untuk dipelajari karena banyak karakter yang mirip satu dengan yang lainnya. Pada penelitian ini digunakan karakter tulisan tangan hiragana dan katakana dasar, dakuten, handakuten, dan youon yang diambil dari responden menggunakan angket. Penelitian ini digunakan metode CNN yang akan dibandingkan dengan gabungan metode CNN dan SVM yang telah dirancang untuk mengenali setiap karakter yang telah dipersiapkan. Preproses gambar karakter menggunakan metode mengubah ukuran gambar, grayscaling, binarisasi, dilasi,dan erosi. Hasil preproses akan menjadi masukan untuk CNN sebagai alat ekstraksi ciri dan SVM sebagai alat untuk pengenalan karakter. Hasil dari penelitian ini sendiri didapat akurasi dengan parameter sebagai berikut ukuran gambar 69×69, nilai patience 3, callbacks monitor val_loss, fungsi optiimasi Nadam, nilai learning rate 0.001, nilai epochs 30, dan kernel SVM rbf. jika menggunakan sistem yang hanya menggunakan jaringan CNN akurasi yang didapat sebesar 87,82%. Hasil yang didapat jika menggunakan gabungan CNN dan SVM sebesar 88,21%.

Kata kunci— *Character-Recognition*, SVM, CNN, *Deep-Learning, Japanese Character Recognition*

Abstract

Hiragana and katakana handwritten characters are often used when writing words in Japanese. Japanese itself is often used by native Japanese as well as people learning Japanese around the world. Hiragana and katakana characters themselves are difficult to learn because many characters are similar to one another. In this study, hiragana and basic katakana, dakuten, handakuten, and youon were used, which were taken from the respondents using a questionnaire. This study used the CNN method which will be compared with a combination of the CNN and SVM methods which have been designed to identify each character that has been prepared. Preprocessing of character images uses the methods of image resizing, grayscaling, binarization, dilation, and erosion. The preprocessed results will be input for CNN as a feature extraction tool and SVM as a tool for character recognition. The results of this study obtained accuracy with the following parameters: 69×69 image size, 3 patience value, val_loss monitor callbacks, Nadam optimization function, 0.001 learning rate value, 30 epochs value, and SVM rbf kernel. If using a system that only uses the CNN network, the accuracy is 87.82%. The results obtained when using a combination of CNN and SVM were 88.21%.

Keywords— *Character-Recognition*, SVM, CNN, *Deep-Learning, Japanese Character Recognition*

Received May 24th,2021; Revised July 31th, 2021; Accepted July 31th, 2021

1. INTRODUCTION

Character recognition is transforming a language represented in a spatial form from a graphic sign into a symbolic representation [1]. In this research a handwritten character of hiragana and katakana are used. The handwriting used can be in the form of handwriting on paper or an object that is digitized using camera or scanner, or online handwriting using a finger or pointer such as a mouse and stylus. There are three types of japanase character namely hiragana, katakana, and kanji and each of them have a writing rule. Some people ignore this rule as long as their writing looks similar to the characters. This makes the Japanese character written by each person are sometimes different.

Some studies have been conducted to create a tool that can recognize hiragana and katakana handwritten characters. Research conducted by Charlie Tsai [2] uses CNN to recognize hiragana, katakana, dan kanji characters. The data used in the study came from the Electronical Laboratory (ETL) Character. ETL data used include ETL-1 for katakana letters, and ETL-8 for hiragana and kanji letters. The result of this study obtainedan accuracy of 99.30% for script recognition, 96.50% for hiragana letter recognition, 98,19% for katakana letter recognition, 99.6% for kanji letter recognition, and 99.53% for all data recognition.

Another research conducted by Wang, et al.[3] who compared the VGG19 algorithm, the CNN algorithm, and the combined CNN-SVM algorithm for charater recognitionof the New Tai Lue letters. The New Tai Lue character data consists of 83 letters, the whole letter data is then normalized to one size which is the same 48×48. Character data obtained from the android applicationwith a total data of 58,795 letters with a total data of 9,834 letters as test data. The results of this research obtained an accuracy 90.66% for VGG19 algorithm with a computation time of 346 seconds, 96.41% for CNN algorithm with a computation time of 197 seconds, and 96.89% for a combined CNN-SVM algorithm with a computation time of 206 seconds.

Darmatasia and Fanany [4] conducted research on handwriting recognition on document entries using the combined CNN and SVM algorithms. The data used were obtained from NIST SD 19 2nd edition, where 1000 samples of data were randomly selected. The selected data is then divided 80% as training data and 20% as testing data. Retrieval of data using the filling form was also carried out on 10 respondents for additional test data. The next step is preprocessing such as segmentation, cropping, and resizing character data to a size of 28×28 pixels. The results of the preprocess are then used as input to the CNN algorithm as a feature extraction tool and produce 192 features that will be predicted in the SVM model. Character prediction testing is done by dividing the data into 4 categories, namely numeric characters, uppercase characters, lowercase characters, and a combination of numeric and uppercase characters. The results of this study reveal that the combined use of the CNN and SVM algorithms in this problem gets better accuracy than when using CNN alone. The best accuracy results obtained are 98.85% numeric characters, 93.05% uppercase characters, 86.21% lowercase characters, and a combination of numeric and uppercase characters 91.37%.

Research conducted by Mane and Kulkarni [5] used CNN for Marathi letter recognition. Modifications were made to the filter section of CNN and the CPU and GPU were used so that the computation process could be faster. Dehghanian and Ghods [6] conducted research using CNN for Farsi letter recognition. Two kinds of experiments were carried out, namely using complete data that had been resized to a size of 28×28 and data with a size of 14×28. The results for full-sized data obtained an accuracy of 98.83% with a computation time of 32.9 seconds and error rate 1.07% and for half-sized data obtained an accuracy of 97.38% with a computation time of 10.16 seconds and error rate 2.62%. So the use of half-sized data makes computing faster, it's just that by using half-sized data there are still recognition errors for numbers 5 and 8

and numbers 4 and 6.

The recognition of Comenia letters using CNN was carried out by Rajnoha et al. [7]. Handwriting samples of Comenia letters were obtained from elementary school children in grades 1 and 2, because elementary school children who had just learned to write letters could not write letters of a similar height so that their recognition would be more difficult. New data were also generated using the skewing, rotation, horizontal flip and vertical flip methods. The use of the Keras and Theano frameworks increases the speed of computing using the GPU. The results obtained were 90.04% for Comenia letter recognition using CNN Deep learning. In the future it is necessary to add more datasets and image quality.

Another research was conducted by Kim and Xie [8] for the pattern recognition of Hangul letters using CNN. The dataset used is SERI95a and PE92. The activation function on CNN is used as follows, the convolutional layer uses the identity function, the max-pooling layer uses the rectified linear function, the classification layer (hidden layer) uses the rectified layer, the classification layer (output layer) uses tanh or softmax. Obtained an accuracy of 95.96% for SERI95a data and 92.92% for PE92 data where there was an increase of 2.25% and 5.22%, which in previous studies the accuracy was 93.71% for SERI95a data and 87.70% for data. PE92.

Suryani et al. [9] conducted a study using a combined CNN and bidirectional long short-term memory (BLSTM) for handwriting recognition of Chinese characters. Character data were obtained from the Chinese Academy of Sciences (CASIA) and used HWDB 1.0-1.2 and HWDB 2.0-2.2. The test data was taken from ICDAR 2013 Handwriting Chinese Character Recognition, where the authors at CASIA and ICDAR 2013 were not the same. This study has the advantage that the initialized CNN filter is not used randomly so that its performance is better. The test used the HWDB 1.1 and ICDAR 2013 dataset. The results obtained 92.00% accuracy for the HWDB 1.1 data set and 92.20% accuracy for the 2013 ICDAR data set.

Research on the pattern recognition of Japanese hiragana, katakana and kanji characters using the Deep Convolutional Recurent Network (DCRN) was conducted by Ly et al. [10]. The model used in this study consists of three layers, namely a convolutional feature extractor CNN and a sliding window to extract data, a recurrent layer containing the Bidirectional Long Short Term Memory (BLSTM) algorithm to predict the label of each feature, a transcription layer containing the CTC algorithm to organize the possible output of the label. The results in the CNN model that the accuracy rate obtained by the validation set is 97.6% and the testing set is 95.17%. Accuracy measurements are also carried out using their performance, namely Label Error Rate (LER) and Sequence Error Rate (SER). The measurement is done by reducing several parts in the model used. The first is a reduction in the softmax (DCRN-s) section, and secondly by reducing the fully-connected layer and softmax (DCRN-f & s) parts, the third is not by using DCRN but using the Segmentation Base. The error rate test carried out produced the following results: DCRN-s, the LER test set value is 6.44% and the SER test set value is 25.89%, the DCRN-f & s test set value is 6.95% LER and the test value The SER set was 28.04%, while for the Segmentation Based the LER test set value was 11.2% and the SER test set value was 48.63%.

In this study the method proposed combination between CNN and SVM for handwriting recognition hiragana and katakana characters. A comparison was made between the character recognition method using CNN and the combined character recognition method of CNN and SVM. This comparison is made to see which method works better between CNN and combine CNN and SVM.

2. METHODS

In this section are explained the data and methods used in this research.

2.1 Datasets

The data used in this research is data handwriting hiragana, and katakana. Data were obtained from respondents using a questionnaire from class XI in the language major of BOPKRI I Yogyakarta Senior High School for the 2019/2020 school year. The following is an example of a questionnaire used in this research.

Figure 1 Example of a questionnaire

In Figure 1, it can be seen that the respondent wrote the letters according to the example in the front column 3 times for each letter.

From the questionnaire that has been collected, the data is processed and cropped using the GIMP application so that the data per character of the desired letter is obtained. An example of the data per letter character that has been obtained is shown in Figure 2

Figure 2 Example of cropped data

There are 104 hiragana letter categories and 122 katakana letter categories, where from the questionnaire that has been collected it is separated again from each letter that has been written 2 letter characters as training data and 1 letter character as test data. So for training data there are (104 × 23 × 2) + (122 × 23 × 2) = 10,396 letters and from the training data 10% is taken for validation data, leaving 9,356 letter data for training and 1,040 letter data for validation data. The test data itself is (104 × 23) + (122 × 23) = 5,198 letters.

2. 2 Preprocessing

First, we preprocess the data we have been resized, similar to that of Sahariar et al. [11] but there are some differences. The preprocess includes resizing the image, grayscale the image, image inversion, image binarization, dilation, and erotion. The read image is then resized as desired in this case 69 × 69. The next step is to convert the color image into a gray image and invert it, which is then converted again into a black and white image through the binarization process. After converting it to black and white image, dilation and erosion are carried out. The dilation process is carried out to increase the area of the existing object. Furthermore, the erosion process is carried out to reduce the area of the object. Images that have been processed through the previous processes are converted into a 69 × 69 × 1 array using the reshape function in the numpy library. All image data that has been processed are collected in an array called x_train for training data and x_test for testing data.

2. 3 Handwriting Recognition System

The system design that will be tested in this study uses CNN and SVM for classification tools. The design will combine the methods drawn from the research literature study conducted by Wang et al. [3] and Ly et al. [10] so that they can find the optimal design both in terms of performance and computational consumption. The CNN algorithm itself can be used for handwriting pattern recognition so that the accuracy of handwritten character recognition will be compared if the program only runs up to CNN with programs running up to CNN and continues to SVM so that it combines CNN and SVM. Figure 3 is a plan for a program that runs up to CNN only.

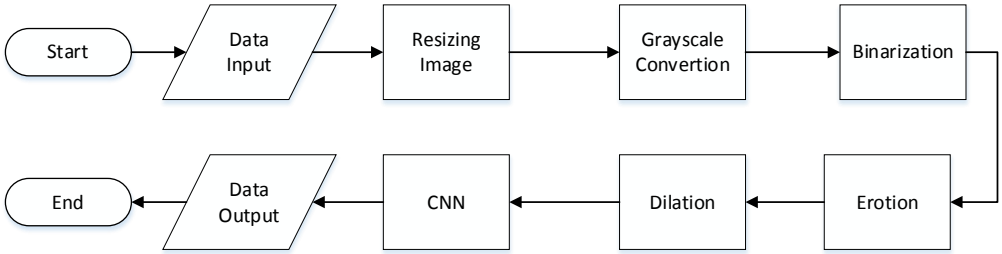

Figure 3 CNN program overview

The steps that will be taken after data collection and scanning the data into a digital image will be converted from an RGB image to a grayscale image. The next step is to change the grayscale image to a black and white image. After becoming a black and white image, the black and white image data will be checked whether the image requires erosion or dilation. This condition is determined by seeing whether there is a lot of noise in the image. The processed data will be entered into input data for the CNN architecture. On the CNN network, an recognition to the character data that has been input is carried out.

The advanced stage of the program that has been designed in Figure 3 is the addition of an SVM for character recognition where the results of training and CNN testing that have been obtained are used as input for SVM. The following is an advanced program design from Figure 3 which has been combined with SVM as shown in Figure 4.

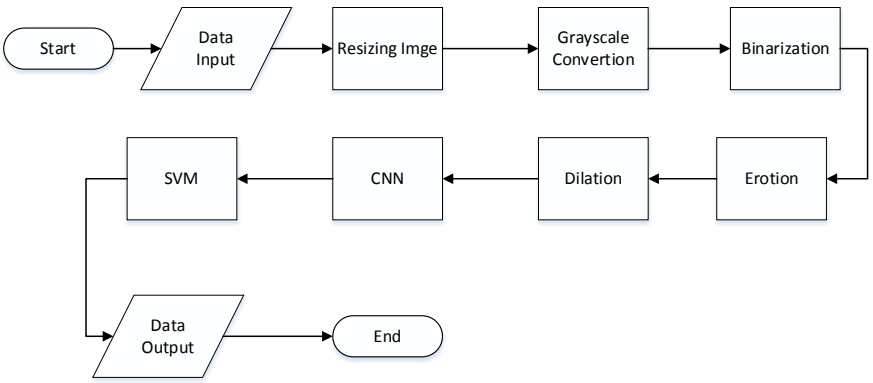

Figure 4 Combined CNN-SVM program overview

In Figure 4, it can be seen that the initial design is similar to Figure 3 but there is an additional SVM as a tool for classification and both designs use the same CNN network.

2.3.1 Convolutional Neural Network

In this research, the CNN network is used and there are 2 main functions that are carried out, namely fire_incepter and fire_squeezer. The flowchart of fire_incepter as shown in figure 5.

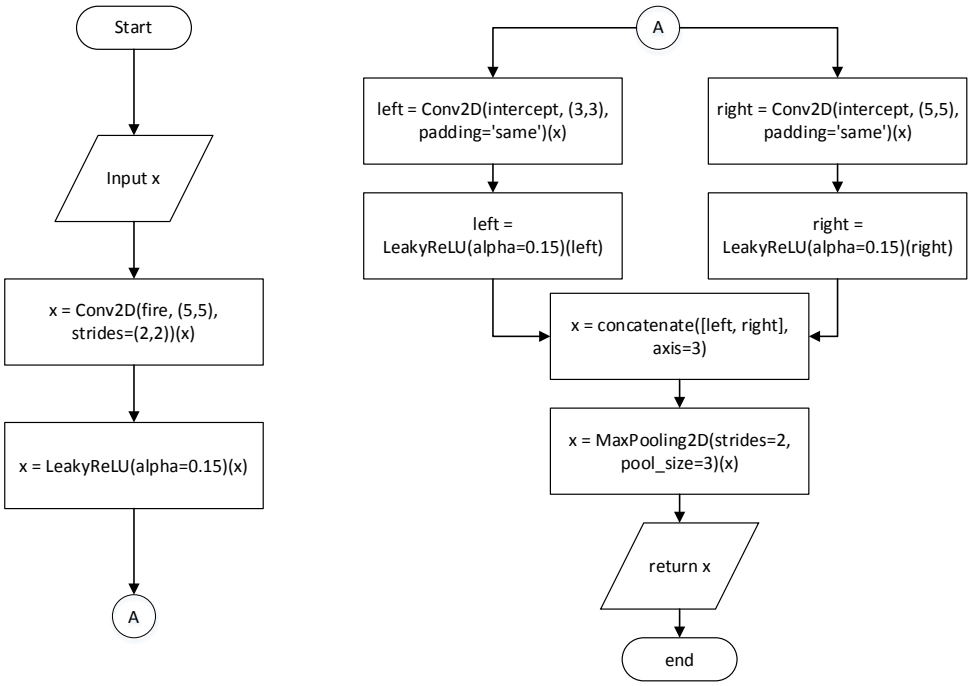

Figure 5 Fire inceepter flowchart

The input data for this function is the variable x which is then convoluted using a convolution layer with a size of 5 × 5 and a shift of 2 × 2. The results from this convolution layer are then inserted into the LeaklyReLu layer with an alpha value of 0.15. Furthermore, the results from the leaklyReLu layer are processed in two different places where x will be convoluted on the left and right. The processing on the left uses a convolution layer with a size of 3 × 3 and which is then stored in the left variable for processing on the LeaklyReLU layer with an alpha value of 0.15. The processing on the right uses a convolution layer with a size of 5 × 5 and which is then stored in the right variable to be processed on the LeaklyReLU layer with an alpha value of 0.15. The results of the left and right variables are then combined and inserted back into the variable x. The result of the combined process is then subjected to MaxPooling with a shift of 2 and a size of 3.

The fire squeezer function itself is similar to the fire incepter function, except that there are some changes to its parameters. The input data for this function is the variable x which is then convoluted using a convolution layer with a size of 16 × 16 and a shift of 1 × 1. The results from this convolution layer are then inserted into the LeaklyReLu layer with an alpha value of 0.15. Furthermore, the results from the leaklyReLu layer are processed in two different places where x will be convoluted on the left and right. The processing on the left uses a convolution layer with a size of 1 × 1 and which is then stored in the left variable for processing on the LeaklyReLU layer with an alpha value of 0.15. The processing on the right uses a convolution layer with a size of 3 × 3 and which is then stored in the right variable to be processed on the LeaklyReLU layer with an alpha value of 0.15. The results of the left and right variables are then combined and inserted back into the variable x. The result of the combined process is then subjected to MaxPooling with a shift of 2 and a size of 2.

The two functions are then called on the CNN model created. The flowchart of the CNN model created can be seen in Figure 6.

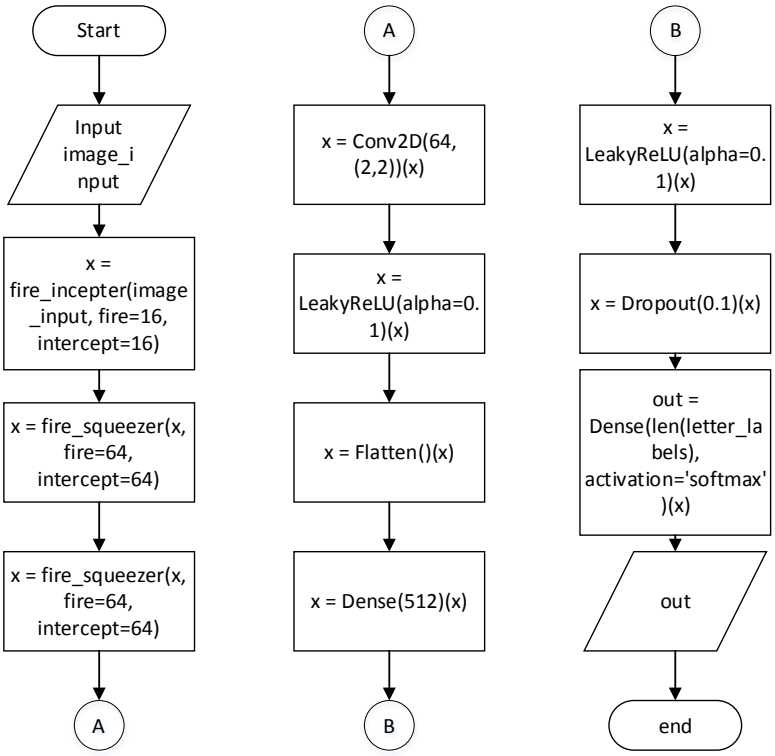

Figure 6 CNN model flowchart

Input is in the form of handwritten character image data with a size of 69 × 69, then the handwritten character data is entered into the fire incepter function which is called on the CNN network. The output from the fire incepter is inputted to the first fire squeezer function called on the CNN network. The second fire squeezer function is called with input data in the form of the output of the first fire squeezer function. The results of the second fire squeezer function are then processed on a convolution layer with a kernel size of 64 and a shift of 2 × 2. The next layer receives input from the convolution layer, namely the LeaklyReLU layer with an alpha value of 0.1. The processed data is then put into the flatten layer and then put into the dense layer with a unit value of 512.The result is then entered into the LeaklyReLU layer with an alpha value of 0.1, which is then added to the Dropout layer with a rate of 0.1. The final part of the CNN is the dense layer with the unit value according to the number of character classes and the softmax activation function.

2. 4 Support Vector Machine

The next step after using CNN is the use of SVM for handwritten character recognition. The following is the SVM design flowchart shown in Figure 7.

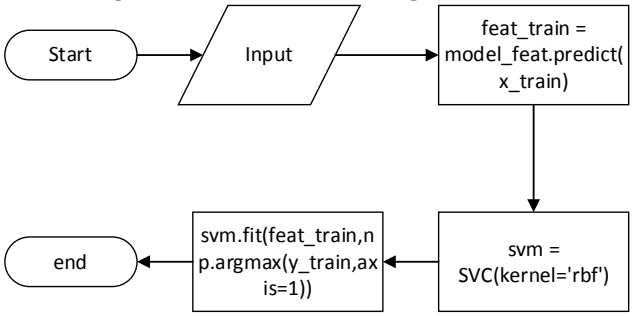

Figure 7 SVM Flowchart

Input data for the created SVM program, taken from the dense_1 layer on the CNN network. The next step is to predict the x_train data. Initialize the SVM kernel which will be used for character recognition. Then do the fitting to the SVM model that has been made.

3. RESULTS AND DISCUSSION

This research focuses on the accuracy results of the CNN algorithm with the combined CNN and SVM algorithms. The test is carried out in two stages. The first stage of testing uses non-hyper parameter variables to find the image size, patience value, and callback functions, and the right SVM kernel. The second stage of testing uses the hyper parameter variable to find the activation function on CNN, the learning rate value, and the correct epochs value.

3. 1 Test Results Using Non Hyper Parameter Variables

In this section, testing is carried out using non-hyper parameter variables such as image size (img_size), patience values on CNN networks, callback functions (callbacks_monitor), and SVM kernel. This is done to obtain a non-hyper parameter variable suitable for testing using the hyper parameter variable. The hyper parameter variable used in testing this non-hyper parameter variable uses the Nadam optimization function, with a learning rate of 0.001 and an epochs value of 30. Table 1 shows the results of CNN network testing using parameters such as image size, number of patience, and callback functions.

Table 1 CNN Test Results Using Non Hyper Parameter Variables

Img _size	patience	callbacks_ monitor	CNN Training Accuration (%)	CNN Validation Accuration (%)	CNN Testing Accuration (%)
63	3	val_loss	94,88	88,85	86,23
63	3	val_accuracy	95,10	87,98	87,23
63	0	val_loss	88,66	81,92	79,90
63	0	val_accuracy	89,94	84,42	83,69
66	3	val_loss	94,73	85,10	84,63
66	3	val_accuracy	93,07	83,17	82,90
66	0	val_loss	91,92	83,08	82,07
66	0	val_accuracy	91,32	83,65	83,86
69	3	val_loss	**96,47**	**89,23**	**88,26**
69	3	val_accuracy	95,15	87,21	87,25
69	0	val_loss	88,13	81,35	80,15
69	0	val_accuracy	91,29	83,17	82,36
72	3	val_loss	94,59	84,62	85,34
72	3	val_accuracy	95,16	85,00	84,97
72	0	val_loss	93,14	84,71	83,40
72	0	val_accuracy	90,62	81,83	82,34

In table 1, we can see some changes in the non-hyper variable parameters used, such as image data sizes ranging from 63 × 63, 66 × 66, 69 × 69 and 72 × 72. The next parameter is the patience value 3 and 0, and then the callback function with the val_accuracy and val_loss monitor callbacks. The highest results from the use of the CNN network for training data were 96.47%, for validation data were 89.23% and 88.26%. The variables used are img_size 69, patience 3, and callbacks_monitor val_loss. The next test uses the feature extraction results from the CNN network that has been made to be included in the SVM model. The results of the SVM used training data with traits extracted via the CNN network. The following are the results of the CNN test followed by SVM as a classification tool which is presented in table 2.

Table 2 SVM Test Results Using Non Hyper Parameter Variables

Img _size	patience	callbacks_monitor	SVM Kernel	SVM Training Accuration (%)	SVM Validation Accuration(%)	SVM Testing Accuration (%)
69	3	val_loss	rbf	**98,46**	**89,71**	**88,76**
69	3	val_loss	linear	98,24	89,33	88,63
69	3	val_loss	poly (degree 2)	98,16	87,40	86,67
69	3	val_loss	poly (degree 3)	97,45	87,12	84,28
69	3	val_loss	sigmoid	97,99	89,33	88,38

The highest results from using the SVM model for training data were 98.46%, for validation data were 89.71% and 88.76%. The variables used are img_size 69, patience 3, callbacks_monitor val_loss and the SVM rbf kernel. The rbf kernel itself in SVM has been set as the default kernel if we use SVM and is a kernel that is often used in the SVM classification.

3.2 Test Results Using Hyper Parameter Variables

In tables 1 and 2, CNN and CNN testing were carried out followed by SVM so that non hyper parameter variables were obtained. These variables are img_size with a value of 69, patience value 3, callbacks_monitor val_loss, and the SVM rbf kernel. Furthermore, the test is carried out using hyper parameters that have been prepared and to be changed, including the optimization function on the CNN network, the learning rate value on the CNN network and the epochs value on the CNN network. Table 3 shows the CNN network testing using fixed non-hyper parameter variables with the img_size 69 value, the patience value 3, the val_loss callback function and the hyper parameter variables tested.

Table 3 CNN Test Results Using Hyper Parameter Variables

Optimizer	learning_rate	epochs	CNN Training Accuration (%)	CNN Validation Accuration (%)	CNN Testing Accuration (%)
Adam	0,01	30	0,61	0,67	0,63
Adam	0,001	30	96,19	87,21	87,03
Adam	0,0001	30	97,76	83,17	83,65
Adamax	0,01	30	92,42	86,15	86,19
Adamax	0,001	30	98,31	85,38	85,44
Adamax	0,0001	30	87,49	74,42	72,76
Nadam	0,01	30	0,48	0,29	0,44
Nadam	0,001	30	**97,37**	**90.00**	**87,82**
Nadam	0,0001	30	95,76	80,29	81,2
Adam	0,01	100	0,44	0,48	0,25
Adam	0,001	100	94,6	87,21	86,42
Adam	0,0001	100	97,49	81,44	81,8
Adamax	0,01	100	90,98	81,92	80,8
Adamax	0,001	100	97,27	85,29	84,71
Adamax	0,0001	100	88,8	73,27	73,8
Nadam	0,01	100	0,59	0,29	0,44
Nadam	0,001	100	95,72	85,58	86,69
Nadam	0,0001	100	97,26	83,37	82,9
Adam	0,01	200	0,86	0,87	1.00
Adam	0,001	200	95,44	87,98	86,28
Adam	0,0001	200	95,88	80,29	79,95
Adamax	0,01	200	93,83	84,71	82,99
Adamax	0,001	200	97,89	81,44	82,32
Adamax	0,0001	200	93,3	76,54	76,53
Nadam	0,01	200	0,49	0,77	0,46
Nadam	0,001	200	95,52	82,69	82,9
Nadam	0,0001	200	95,86	80,48	79,55

Table 3 shows the use of variable hyper parameter variables such as the optimization function using Adam, Nadam, and Adamax. The next variable is the learning rate values of 0.01, 0.001, and 0.0001. Furthermore, changes in the value of epochs with values of 30, 100, and 200. It can also be seen that changes in hyper parameters on the CNN network itself can provide quite a difference between one variable and another. The highest result was for the CNN network with an accuracy of 97.37% for training data, 90.00% for validation data, and 87.82% for testing data. The variables used to obtain these results are the Nadam optimization function, the learning rate value of 0.001, and the epochs value of 30. Furthermore, testing is carried out using SVM with the rbf kernel that has been determined from the results of the previous test for the non-hyper parameter variable which is presented in table 4.

Table 4 CNN Test Results Using Hyper Parameter Variables

Optimizer	learning_rate	epochs	SVM Kernel	SVM Training Accuration (%)	SVM Validation Accuration (%)	SVM Testing Accuration (%)
Nadam	0,01	30	rbf	4,77	3,37	3,14
Nadam	0,001	30	rbf	**98,44**	**89,71**	**88,21**
Nadam	0,0001	30	rbf	99,49	80,19	81,44

Table 4 shows that the highest accuracy obtained is 98.44% for training data, 89.71% for validation data and 88.21% for testing data. There is a slight difference in the accuracy value from that obtained when testing with non hyper parameter variables and when testing with hyper parameter variables. This is because the validation data collection takes 10% of the training data randomly.

4. CONCLUSIONS

After the research was carried out, the research results were obtained with the following conclusions:

The experiments that have been carried out are divided into two, the first is an experiment by changing the non-hyper parameter variables, among others, by changing the image size with values of 63 × 63, 66 × 66, 69 × 69, and 72 × 72. Changing experiments were also carried out on the CNN model by changing the patience value of the callback function by 0 or 3, changing the monitor callbacks parameter in the callback function to val_loss or val_accuracy. In the SVM model the experiment was carried out using the rbf kernel, linear, poly with degree 2, poly with degree 3, and sigmoid.

The results of testing experiments with non-hyper variable parameters, the greatest accuracy rate is obtained using image size parameters 69 × 69, with patience value 3, monitor callbacks val_loss, and SVM kernel using rbf. These parameter parameters then become fixed parameters to be tested in the test using the hyper parameter variable. The highest accuracy results obtained were 98.46% for training data, 89.71% for validation data, and 88.76% for testing data.

Subsequent experiments were carried out by changing the hyper parameter variables including the optimization function on the CNN network with Adam, Nadam, or Adamax. Other changes change the value of the learning rate used on the CNN network with a value of 0.01, 0.001, or 0.0001. Furthermore, the epochs value changes to 30, 100, or 200.

The results of testing experiments with hyper parameter variables, the greatest accuracy rate is obtained using the parameters of the Nadam optimization function, with a learning rate of 0.001, and an epochs value of 30. Parameters parameters The highest accuracy result obtained is 98.44% for training data, 89.71% for validation data and 88.21% for testing data.

The strength of this research is the combination of the CNN algorithm and the SVM algorithm for handwriting recognition of hiragana and katakana characters. It is also known that the combined use of the CNN algorithm and the SVM algorithm can run better than the use of the stand-alone CNN algorithm. Another advantage is that it is known that non hyper parameter variables and hyper parameter variables are used with high accuracy so that further research can use these variables. The drawback of this study is the lack of handwritten data on hiragana and katakana characters in Indonesia, so that the respondents are not taken from native Japanese people but students who studied Japanese.

The conclusion is using the CNN network structure that has been created, there is an average difference in the accuracy of 1% -1.5% when training data, validation data, and test data are used in the CNN algorithm followed by SVM. This difference shows that the combined algorithm from CNN + SVM can be better than the stand-alone CNN algorithm.

5. SUGGESTION

Suggestions for further research should be added to the training data so that the tools that have been made can produce higher accuracy. The training data may also be better if taken from the original author, who is a native Japanese or a person who has lived in Japan for a long time. The image processing process also needs to be considered so that the characteristic data obtained is better. Addition or change in the structure of the artificial neural network so that the characterization is better and the accuracy is higher.

REFERENCES

[1] V. Kumar, "Online Handwriting Recognition Problem : Issues and Techniques," vol. 4, no. 1, pp. 16–24, 2014.

[2] Charlie Tsai, "Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks," pp. 1–7, 2016.

[3] Y. Wang, P. Yu, and C. Li, "Offline Handwritten New Tai Lue Characters Recognition Using CNN-SVM," *Proc. 2019 IEEE 2nd Int. Conf. Electron. Inf. Commun. Technol. ICEICT 2019*, pp. 636–639, 2019, doi: 10.1109/ICEICT.2019.8846292.

[4] Darmatasia and M. I. Fanany, "Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM)," *2017 5th Int. Conf. Inf. Commun. Technol. ICoIC7 2017*, vol. 0, no. c, pp. 1–6, 2017, doi: 10.1109/ICoICT.2017.8074699.

[5] D. T. Mane and U. V Kulkarni, "Visualizing and Understanding Customized Convolutional Neural Network for Recognition of Handwritten Marathi Numerals," *Procedia Comput. Sci.*, vol. 132, no. Iccids, pp. 1123–1137, 2018, doi: 10.1016/j.procs.2018.05.027.

[6] A. Dehghanian and V. Ghods, "Farsi Handwriting Digit Recognition based on Convolutional Neural Networks," *2018 6th Int. Symp. Comput. Bus. Intell.*, vol. 1, pp. 65–68, 2018, doi: 10.1109/ISCBI.2018.00022.

[7] M. Rajnoha, R. Burget, and M. Khisore Dutta, "Handwriting Comenia Script Recognition with Convolutional Neural Network," pp. 775–779, 2017.

[8] I.-J. Kim and X. Xie, "Handwritten Hangul recognition using deep convolutional neural networks," pp. 1–13, 2015, doi: 10.1007/s10032-014-0229-4.

[9] D. Suryani, P. Doetsch, and H. Ney, "On the Benefits of Convolutional Neural Network Combinations in Offline Handwriting Recognition," *2016 15th Int. Conf. Front. Handwrit. Recognit.*, pp. 193–198, 2016, doi: 10.1109/ICFHR.2016.0046.

[10] N. T. Ly, C. T. Nguyen, K. C. Nguyen, and M. Nakagawa, "Deep Convolutional Recurrent Network for Segmentation-Free Offline Handwritten Japanese Text Recognition," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, vol. 7, pp. 5–9, 2018,

doi: 10.1109/ICDAR.2017.357.

[11] A. Shahariar, A. Rabby, S. Haque, S. Islam, S. Abujar, and S. A. Hossain, "BornoNet : Bangla Handwritten Characters Recognition Using Convolutional Neural Network Convolutional Neural Network," *Procedia Comput. Sci.*, vol. 143, pp. 528–535, 2018, doi: 10.1016/j.procs.2018.10.426.