[1] **Dwi Arman Prasetya**
[2] **Mohammad Idhom**
[3] **Anggraini Puspita Sari**
[4] **Irma Amanda Putri**
[5] **Adinda Aulia Rahmawati**

Implementation of Convolutional Neural Network for Hiragana Classification

Journal of Electrical Systems

Abstract: - Hiragana is one of three writing systems in Japanese that is widely used in various texts, from literature to everyday media. Hiragana characters consist of 46 basic symbols, each representing one syllable. Hiragana character classification is a complex challenge in image processing and machine learning. Despite various advances, achieving high accuracy in Hiragana character classification remains difficult. This research aims to increase the accuracy of Hiragana character classification by applying a Convolutional Neural Network (CNN). The CNN method was chosen because of its strong feature extraction and image classification capabilities. This research focuses on the optimizers, including Adam, SGD, RMSprop, Adamax, Adagrad, and Adadelta, for Hiragana character classification. The dataset used in this research focuses on ten Hiragana characters: Aa, Ki, Su, Tsu, Na, Ha, Ma, Ya, Re, and Wo. The research results show that using the Adam optimizer in the CNN model training process achieved the highest accuracy of 97.37%. Model performance analysis also considers important metrics such as precision, recall, and F1-Score to validate the effectiveness of the Adam optimizer in improving model performance. These results emphasize the role of the Adam optimizer in improving the accuracy of image classification models. This research significantly contributes to Hiragana character classification by highlighting the importance of choosing the right optimizer to improve model performance. In conclusion, the Adam optimizer selection proved effective in increasing the accuracy of Hiragana character classification using CNN.

Keywords: Classification, Hiragana, Convolutional Neural Network, Optimizer

I. INTRODUCTION

Various communities and countries. Each language has its own uniqueness and characteristics that reflect the culture and history of the people who use it [1]. Among these languages, Japanese stands out for its unique and complex writing system, which includes Hiragana, Katakana, and Kanji. Japanese is a language used by many people, especially in Japan, but also by Japanese communities throughout the world [2].

The Japanese language attracts the attention of many people for several reasons. Japan is one of the largest economies in the world and a center for technological innovation, so the ability to speak Japanese can open up career and business opportunities [3]. Additionally, Japan has a rich and deep cultural heritage, which is reflected in art, literature and traditions. The popularity of Japanese pop culture such as anime, manga and music is also a big attraction, making many people interested in learning this language in order to understand these works in their original language.

Hiragana is one of the three main writing systems in Japanese, which also includes Katakana and Kanji[4]. Each Hiragana character represents one syllable, and this system is very important in the learning and everyday use of Japanese. Due to its phonetic nature and the relatively small number of characters compared to Kanji. Hiragana is frequently used in children's literature, everyday media, and educational texts [5]. Hiragana consists of 46 basic symbols, covering all the basic sounds in Japanese [6].

Hiragana is very important in basic education in Japan. Japanese children start learning Hiragana as early as kindergarten, as it is the foundation for reading and writing [7]. Additionally, for foreign students studying Japanese, mastering Hiragana is an essential first step before moving on to katakana and kanji [8]. Learning Hiragana helps understand the basic structure of the Japanese language and makes it easier to read material written in Japanese, be it books, newspapers or websites. In this context, one of the problems that arises is the difficulty in recognizing and classifying Hiragana automatically. Manual recognition of Hiragana requires deep knowledge and special skills, which not everyone has. Therefore, the use of computational methods like Artificial Intelligence to automatically classify Hiragana becomes very important [9].

Automatic classification of Hiragana characters using image processing and machine learning techniques is a complex challenge. Because some Hiragana characters have very similar shapes, which can cause confusion in the classification process. This research aims to improve the accuracy of Hiragana character classification by applying

[1] Universitas Pembangunan Nasional "Veteran" Jawa Timur, Indonesia. Email: arman.prasetya.sada@upnjatim.ac.id
[2] Universitas Pembangunan Nasional "Veteran" Jawa Timur, Indonesia. Email: idhom@upnjatim.ac.id
[3] Universitas Pembangunan Nasional "Veteran" Jawa Timur, Indonesia. Email: anggraini.puspita.if@upnjatim.ac.id
[4] Universitas Pembangunan Nasional "Veteran" Jawa Timur, Indonesia. Email: irmaamandaputri@gmail.com
[5] Universitas Pembangunan Nasional "Veteran" Jawa Timur, Indonesia. Email: adindaauliarahmawati22@gmail.com

Convolutional Neural Networks (CNN). CNNs were chosen for powerful feature extraction and image classification capabilities [10], [11]. CNNs have proven to be highly effective in a variety of image recognition tasks, including handwriting recognition and object classification [12], [13].

CNNs can recognize basic patterns such as edges and textures in Hiragana images, and have invariance to small shifts, scales, and rotations in images, making them very suitable for handling variations in Hiragana [14]. Through pooling layers, CNN can reduce image dimensionality without losing important information, thereby increasing computational efficiency [15]. With flexible architecture and regulation techniques such as dropout and batch normalization, CNNs are able to reduce the risk of overfitting and increase model accuracy [16].

This research also focuses on comparing various optimizers used in the CNN model training process. Optimizers are important components in machine learning that influence model convergence and performance. The optimizers tested in this research include Adam, SGD, RMSprop, Adamax, Adagrad, and Adadelta. Each optimizer has different characteristics and mechanisms for managing weight updates during training.

Similar research has been carried out previously entitled "Transliteration of Hiragana and Katakana Handwritten Characters Using CNN-SVM". In this study, using CNN, the accuracy was 87.82% [17]. Apart from that, the research entitled "State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images" states that the optimizer has a significant role in the model accuracy results [18].

Based on the existing background and literature in conducting research, CNN can be applied in classifying Hiragana characters. This research is expected to make a significant contribution in the field of Hiragana character classification by exploring optimal CNN architectures, analyzing the performance of various optimizers. By choosing the right optimizer, it is hoped that the accuracy of Hiragana character classification can be improved.

II. METHODS

The methodology used in this research is shown in Fig 1, starting with preparing the Hiragana dataset. The data then goes through a preprocessing stage which includes cleaning, labeling, and creating validation data. After that, a Convolutional Neural Network (CNN) architecture was built and its hyperparameters were tested. The CNN model is trained using processed data, then used for Hiragana classification. The final stage is evaluating the model's performance using a confusion matrix to assess how well the model is at classifying the data.

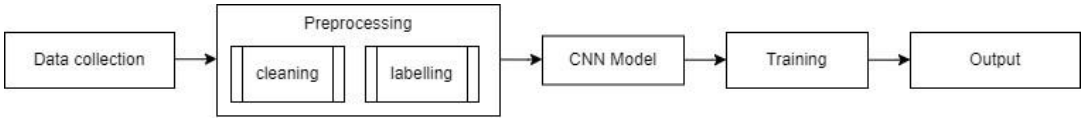

Fig 1. Flowchart

A. Data Collection

The dataset used in this research was taken from the data provider website, namely Kaggle (url: kaggle.com) with the name of the data used being Japanese Characters in Fig 2. This research focuses on 10 hiragana characters, namely Aa, Ki, Su, Tsu, Na, Ha, Ma, Ya, Re, and Wo. The dataset has a total of 70,000 data, for 50,000 train data, 10,000 validation data, and 10,000 test data. The dataset used is in CSV format

Fig 2. Hiragana Character

B. Preprocessing

Preprocessing is a series of steps performed to prepare raw data before it is used in modeling or analysis. In the context of machine learning and natural language processing, preprocessing is critical because the quality of the input data can significantly affect model performance. In this research, preprocessing was carried out, namely cleaning and labeling data.

1. Cleaning Data

One of the steps in data cleaning is deleting columns that are not needed or that do not carry significant information. The 'Unnamed: 0' column often appears when a CSV or Excel file is read into a dataframe, and is usually a default index that is not needed if a real index already exists or if the data has no analytical value. This also helps maintain computational efficiency by reducing the number of columns that need to be processed.

2. Labeling Data

The data labeling process is a crucial step in preparing data for analysis and machine learning model development. The 'label' column contains the target or output that the model wants to predict. The labels used in this research are 0: "Aa", 1: "Ki", 2: "Su", 3: "Tsu", 4: "Na", 5: "Ha", 6: "Ma", 7 : "Ya", 8 : "Re", 9 : "Wo".

C. CNN Architecture

The model architecture used is a convolutional neural network (CNN) in Fig 3 designed for image classification tasks. This model was built using Keras' Sequential API and consists of several main layers. The model starts with an input layer that receives a 28x28 pixel image with 1 color channel (grayscale).

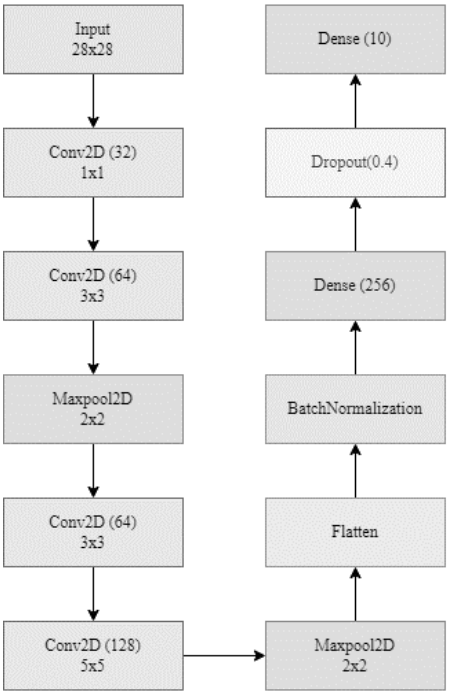

Fig 3. CNNs Architecture

The first layer is Conv2D with 32 filters, 1x1 kernel size, "same" padding, and ReLU activation function. This layer is followed by a second Conv2D with 64 filters and a 3x3 kernel, also with "same" padding and ReLU activation. Next, there is a MaxPooling2D layer with a pool size of 2x2 to reduce the spatial dimensions of the image. After that, there are again two Conv2D layers with 64 and 128 filters respectively, 3x3 and 5x5 kernels, both with "same" padding and ReLU activation. This is followed by a second MaxPooling2D layer with a pool size of 2x2. The next layer is the Flatten layer which flattens the output of the convolution layer into a one-dimensional vector. Next, there is a BatchNormalization layer to normalize the output of the previous layer, which helps speed up training and model stability.

Then, there is a Dense layer with 256 neurons, using L2 and L1 regularization for kernels and bias, as well as a ReLU activation function. This layer also has Dropout with a 40% dropout rate to prevent overfitting. The Dense output layer has a number of neurons corresponding to the number of classes to predict (class_count), with a softmax activation function to generate class probabilities. Overall, this architecture is designed to capture spatial features from images through convolution and pooling layers, followed by fully connected layers to perform final classification based on the extracted features. The following is the CNN architecture used.

D. Training

Training begins by initializing parameters such as 15 epochs based on the amount of test data and batch size, optimization using the Adam, SGD, RMSprop, Adamax, Adagrad, and Adadelta functions. Adam (Adaptive Moment Estimation) combines the advantages of RMSprop and Momentum by using the first and second moments of the gradient to update the weights, which makes it fast and often results in better convergence. SGD (Stochastic Gradient Descent) is a basic optimizer that updates parameters based on the gradient of each example in the dataset, but it can be slow because the direction of the gradient can vary greatly. RMSprop (Root Mean Square Propagation) addresses the problem of slow gradient descent by taking into account the moving average of the squares of recently obtained gradients, ensuring a more stable and fast update step.

Adamax, a variant of Adam, based on maximum norms, provides more stable results on some types of problems. Adagrad (Adaptive Gradient Algorithm) adjusts the learning rate for each parameter adaptively based on previously received gradients, effective for sparse features but tends to slow down over time due to its continuously decreasing learning rate. Adadelta is an improvement on Adagrad that attempts to address the problem of ever-decreasing learning rates by limiting the gradient cumulative window and introducing new parameters to maintain a more constant learning rate.

By using these various optimizers, it is hoped that optimal performance of the model can be achieved, depending on the specific nature of the dataset and the problem at hand. Loss uses sparse categorical crossentropy, metrics such as accuracy, validation data, and other requirements are tailored by each researcher. During this training process, a GPU from Kaggle is utilized to enhance the computing performance.

E. Output

The evaluation stage is the stage of checking the accuracy of the experimental results which are designed into several test scenarios. Evaluation is the stage of measuring the performance of a model that has been produced. At the evaluation stage, this research uses a confusion matrix to measure the performance of an algorithm. Table 1 are the internal components confusion matrix.

Table 1. Confusion Matrix

Confusion Matrix	Predicted Class	
	Positive	Negative
True Class	FN	FP
	TP	TN

Information:
TP = The count of positive instances that are classified accurately
TN = The count of negative instances that are classified accurately
FP = The count of positive instances that are classified incorrectly
FN = The count of negative instances that are classified incorrectly

III. RESULT

The model evaluation results which describe the training and validation accuracy as well as the training and validation loss of the model provide a clearer visual picture of how the model performance develops along with the training and validation process, accuracy and generalization in classifying data. Fig 4 is an evaluation of the Adam optimizer.

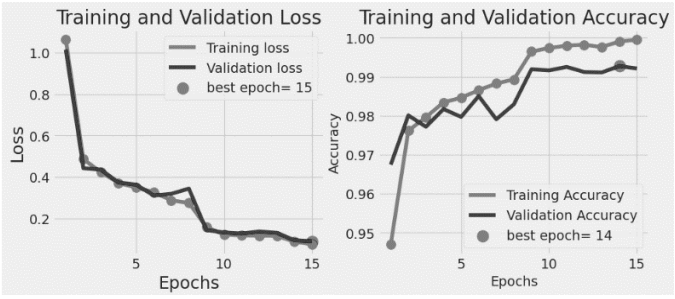

Fig 4. Graph Adam Optimizer

Based on the Fig 4, the first graph shows a significant decrease in loss values for both training and validation data, with the best epoch identified at epoch 15. Initially, both training and validation loss were high but decreased significantly as the epochs increased. The second graph demonstrates a steady increase in accuracy for both training

and validation data, with the best epoch identified at epoch 15. Training accuracy increased steadily and reached nearly 1.0, while validation accuracy also improved despite some fluctuations. Overall, the model shows good performance with adequate generalization and no significant overfitting.

The second optimizer uses Adamax, Fig 5 are the results of the evaluation of the Adamax optimizer.

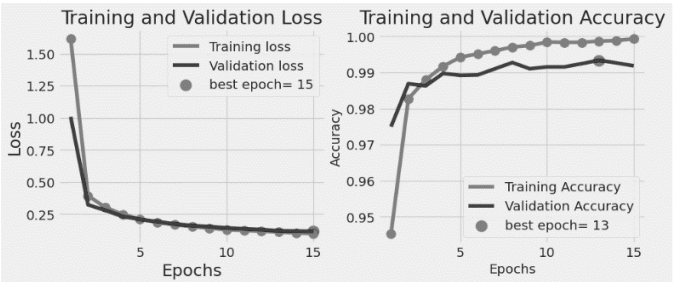

Fig 5. Graph Adamax Optimizer

Based on the Fig 5, the first graph shows a significant decrease in loss values for both training and validation data, with the best epoch identified at epoch 15. Initially, both training and validation loss were very high, around 1.5, but they decreased rapidly as the epochs increased. The second graph demonstrates a steady increase in accuracy for both training and validation data, with the best epoch identified at epoch 15. Training accuracy increased quickly and reached nearly 1.0, while validation accuracy also improved, stabilizing with minor fluctuations.

The third optimizer uses RMSprop, Fig 6 are the evaluation results of the RMSprop optimizer.

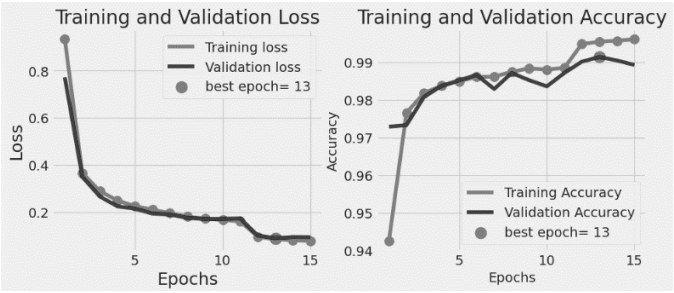

Fig 6. Graph RMSprop Optimizer

Based on the Fig 6, the first graph shows a drastic decrease in the loss values for both training and validation data, with the best epoch being at epoch 13. At the beginning of training, the loss value is very high, around 0.8, but decreases rapidly as the number of epochs increases. The second graph shows a consistent increase in accuracy for both training and validation data, with the best epoch also being at epoch 13. The training accuracy increases rapidly approaching 1.0, while the validation accuracy also increases and stabilizes with little fluctuation.

The fourth optimizer uses Adagrad, Fig 7 are the results of the evaluation of the Adagrad optimizer.

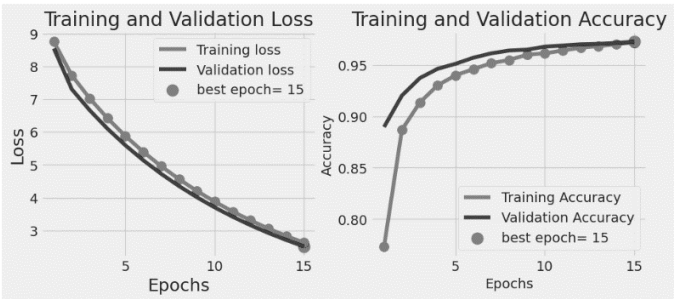

Fig 7. Graph Adagrad Optimizer

Based on the Fig 7, the first graph shows a significant decrease in loss values for both training and validation data, reaching a low point at epoch 15. The rapid decrease in loss early in training indicates that the model is quickly able to pick up patterns in the data. The second graph shows a consistent increase in accuracy for both types of data, with the best performance also achieved at epoch 15. The training and validation accuracy curves are quite close, indicating that the model is able to generalize well to data that it has never seen before. There is no significant indication of overfitting, such as a sharp increase in validation loss or a decrease in validation accuracy after a certain epoch.

The fifth optimizer uses SGD, Fig 8 are the evaluation results of the SGD optimizer.

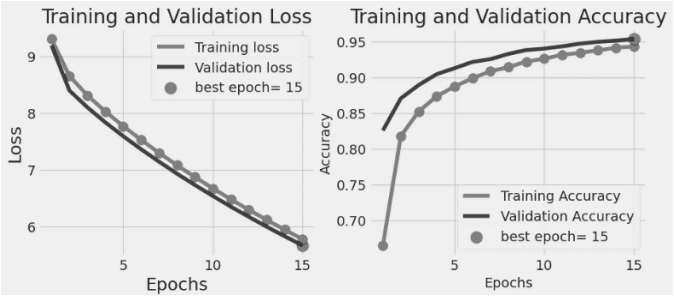

Fig 8. Graph SGD Optimizer

Based on the Fig 8 shown, the trained machine learning model shows very good performance. Both loss and accuracy values, both on training and validation data, show a positive trend. A significant decrease in loss values and a consistent increase in accuracy indicate that the model has successfully learned complex patterns in the data. There is no indication of significant overfitting, as seen from the similarity between the loss and accuracy curves on training and validation data. Epoch 15 is identified as the optimal point where the model achieves the best performance.

The sixth optimizer uses Adadelta, Fig 9 are the results of the evaluation of the Adadelta optimizer.

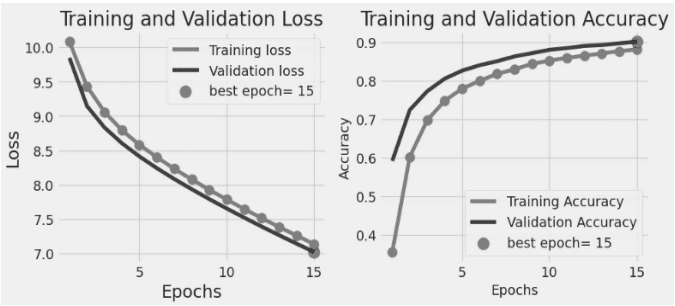

Fig 9. Graph Adadelta Optimizer

Based on the Fig 9, the training results displayed in the graph, the machine learning model has shown excellent performance. The loss value that continues to decrease in both training and validation data indicates that the model is increasingly able to capture complex patterns in the data. In addition, the consistent increase in accuracy also shows that the model is increasingly adept at making accurate predictions. There are no significant symptoms of overfitting, as seen from the similarity between the loss and accuracy curves on both types of data. Epoch 15 is identified as the optimal point where the model achieves the best performance.

Based on the evaluation results in Figure 4 to Figure 9, it can be seen that the six optimizers tested showed quite good performance in reducing the loss value and increasing the model accuracy. However, after conducting a comprehensive comparison, it can be concluded that the Adam optimizer gave the best results in training this model. The Adam optimizer managed to reach the optimal point at the 15th epoch with the lowest loss value and the highest accuracy compared to other optimizers. In addition, Adam also showed good stability in model performance, with loss and accuracy curves that tended to be smoother than other optimizers. Although other optimizers such as Adamax, RMSprop, and Adagrad also showed quite good results, Adam consistently excelled in terms of convergence speed, stability, and overall performance. The SGD and Adadelta optimizers tended to have slower and less stable performance compared to other optimizers. Adam optimizer proved to be an effective choice for this model, producing the best performing model in terms of accuracy and generalization.

After carrying out several experiments using the same architecture and parameters with various optimizers, the resulting accuracy on the test data is as shown in the table 2.

Table 2. Accuracy Each Optimizer

Optimizer	Accuracy (%)
Adam	97.37
Adamax	96.84
RMSprop	96.78
Adagrad	91.57
SGD	86.89
Adadelta	78.07

From the accuracy comparison results of the optimizers above, it is known that Adam (Adaptive Moment Estimation) shows higher accuracy in hiragana classification compared to other optimizers because of its adaptive and efficient capabilities. Adam uses learning rates that adapt to each parameter, allowing more precise updates as needed. Additionally, Adam incorporates momentum concepts to speed up convergence and reduce oscillations, and uses bias correction to maintain the initial stability of parameter updates. Adam's ability to handle rare gradients and complex data variations makes it more robust and effective in optimizing parameters, resulting in higher accuracy in hiragana character classification compared to other optimizers such as SGD, RMSprop, and Adagrad.

Fig 10 is a more detailed confusion matrix value from the Adam optimizer.

	precision	recall	f1-score	support
Aa	0.9656	0.9820	0.9737	1000
Ki	0.9857	0.9670	0.9763	1000
Su	0.9720	0.9360	0.9536	1000
Tsu	0.9565	0.9900	0.9730	1000
Na	0.9637	0.9560	0.9598	1000
Ha	0.9866	0.9590	0.9726	1000
Ma	0.9566	0.9920	0.9740	1000
Ya	0.9899	0.9770	0.9834	1000
Re	0.9764	0.9940	0.9851	1000
Wo	0.9860	0.9840	0.9850	1000
accuracy			0.9737	10000
macro avg	0.9739	0.9737	0.9737	10000
weighted avg	0.9739	0.9737	0.9737	10000

Fig 10. Classification Report Adam Optimizer

Fig 11 is a more detailed confusion matrix from the Adam optimizer.

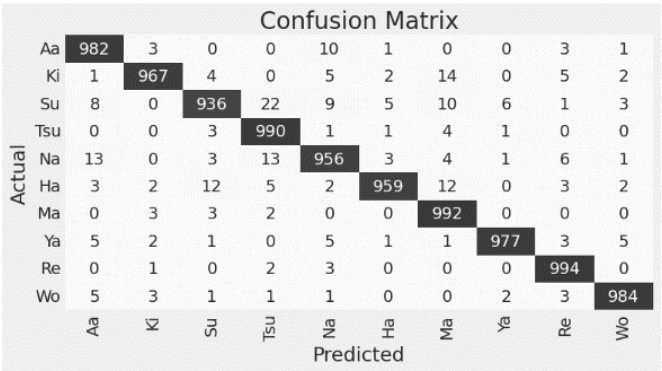

Fig 11. Confusion Matrix Adam Optimizer

IV. CONCLUSION

Based on the research findings, it can be concluded that the Convolutional Neural Network (CNN) method for Hiragana classification achieves an accuracy of 97.37% when using the Adam optimizer, with 50,000 data points for training, 10,000 for validation, and 10,000 for testing. The results from this model are expected to be developed into an engaging application for teaching Hiragana to foreign students, enhancing their learning experience. However, further investigation is needed to refine the accuracy of the validation data. Ultimately, these findings could be utilized for advancing the recognition of sentence sequences in Hiragana.

REFERENCES

[1] D. Kim, "Learning Language, Learning Culture: Teaching Language to the Whole Student," *ECNU Rev. Educ.*, vol. 3, no. 3, pp. 519–541, 2020, doi: 10.1177/2096531120936693.

[2] L. A. Turumbetova, "The impact of globalization on the oriental languages, education and culture," *J. Orient. Stud.*, vol. 93, no. 2, pp. 154–162, 2020, doi: 10.26577/jos.2020.v93.i2.18.

[3] T. S. Adebayo and D. Kirikkaleli, "Impact of renewable energy consumption, globalization, and technological innovation on environmental degradation in Japan: application of wavelet tools," *Environ. Dev. Sustain.*, vol. 23, no. 11, pp. 16057–16082, 2021, doi: 10.1007/s10668-021-01322-2.

[4] Y. Harun and F. N. Biduri, "Historical Analysis of Japanese Writing Systems Hiragana, Katakana, and Kanji," *Int. J. Soc. Serv. Res.*, vol. 4, no. 02, pp. 612–618, 2024, doi: 10.46799/ijssr.v4i02.720.

[5] T. Tanji and T. Inoue, “Home literacy environment and early reading skills in Japanese Hiragana and Kanji during the transition from kindergarten to primary school,” *Front. Psychol.*, vol. 14, no. April, 2023, doi: 10.3389/fpsyg.2023.1052216.
[6] R. Arni and P. Suciaty, “ An Analysis of Students’ Hiragana Letters Mastery at Japanese For General Purpose Course of Universitas Negeri Padang ,” *Proc. 2nd Prog. Soc. Sci. Humanit. Educ. Res. Symp. (PSSHERS 2020)*, vol. 563, no. Psshers 2020, pp. 24–29, 2021, doi: 10.2991/assehr.k.210618.005.
[7] T. Inoue *et al.*, “Reading in different scripts predicts different cognitive skills: evidence from Japanese,” *Read. Writ.*, vol. 35, no. 6, pp. 1425–1448, 2022, doi: 10.1007/s11145-021-10228-4.
[8] S. R. Paxton, “Tackling the Kanji Hurdle: An investigation of kanji order and its role in facilitating the kanji learning process,” no. April, 2022.
[9] D. A. Prasetya, P. T. Nguyen, R. Faizullin, I. Iswanto, and E. F. Armay, “Resolving the shortest path problem using the haversine algorithm,” *J. Crit. Rev.*, vol. 7, no. 1, pp. 62–64, 2020, doi: 10.22159/jcr.07.01.11.
[10] M. Idhom, D. A. Prasetya, P. A. Riyantoko, T. M. Fahrudin, and A. P. Sari, “Pneumonia Classification Utilizing VGG-16 Architecture and Convolutional Neural Network Algorithm for Imbalanced Datasets,” *TIERS Inf. Technol. J.*, vol. 4, no. 1, pp. 73–82, 2023, doi: 10.38043/tiers.v4i1.4380.
[11] A. P. Sari, D. A. Prasetya, T. Yasuno, A. N. Sihananto, M. M. Al Haromainy, and W. S. J. Saputra, “Forecasting Model of Wind Speed and Direction by Convolutional Neural Network - Deep Convolutional Long Short Term Memory,” *Proceeding - IEEE 8th Inf. Technol. Int. Semin. ITIS 2022*, pp. 200–205, 2022, doi: 10.1109/ITIS57155.2022.10010005.
[12] A. A. Rahmawati, A. Muhaimin, and D. A. Prasetya, “Classification of Javanese Nglegena Script Using Complexvalued Neural Network,” *JIKO (Jurnal Inform. dan Komputer)*, vol. 7, no. 1, pp. 30–35, 2024, doi: 10.33387/jiko.v7i1.7808.
[13] I. A. Putri, D. A. Prasetya, and T. M. Fahrudin, “Image Classification of Vine Leaf Diseases Using Complex-Valued Neural Network,” *JIKO (Jurnal Inform. dan Komputer)*, vol. 7, no. 1, pp. 36–42, 2024, doi: 10.33387/jiko.v7i1.7809.
[14] L. Alzubaidi *et al.*, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.
[15] R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang, “Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study,” *Neural Comput. Appl.*, vol. 34, no. 7, pp. 5321–5347, 2022, doi: 10.1007/s00521-022-06953-8.
[16] I. Al-Shourbaji *et al.*, “A Deep Batch Normalized Convolution Approach for Improving COVID-19 Detection from Chest X-ray Images,” *Pathogens*, vol. 12, no. 1, 2023, doi: 10.3390/pathogens12010017.
[17] N. E. W. Nugroho and A. Harjoko, “Transliteration of Hiragana and Katakana Handwritten Characters Using CNN-SVM,” *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 15, no. 3, p. 221, 2021, doi: 10.22146/ijccs.66062.
[18] M. Yaqub *et al.*, “State-of-the-art CNN optimizer for brain tumor segmentation in magnetic resonance images,” *Brain Sci.*, vol. 10, no. 7, pp. 1–19, 2020, doi: 10.3390/brainsci10070427.