INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Handwritten Hiragana Letter Detection Using CNN

Arya Fernandi [a,*], Rita Magdalena [a], Sofia Saidah [a]

[a] *School of Electrical Engineering, Telkom University, Bandung, Indonesia*

Corresponding author: [*]aryafernandi4@gmail.com

***Abstract*—*Hiragana* is one of the primary alphabets used in Japanese. *Hiragana* is a phonetic symbol; each letter represents one syllable. *Hiragana* letters are formed from curved lines and strokes. However, detecting *Hiragana* letters causes many errors because people still rely on their vision to detect the letters, especially people familiar with them for the first time. It will be difficult and not very clear to read the letters. Therefore, a Convolutional Neural Network (CNN) method is used to detect handwritten *Hiragana* letters and help people who first get to know *Hiragana* letters when the letters are too complicated for human eyes to detect. This research uses the YOLOv8 model as a handwritten *Hiragana* letter detection algorithm. The *Hiragana* letters to be detected are basic letters with 46 characters. This research uses the YOLOv8 model run on Google Collaboratory with the Ultralytics library version 8.0.20 using the Python programming language. The dataset is collected from the internet and annotated using the Roboflow framework and dataset 4600 *Hiragana* letters. From the test results, the best model is YOLOv8l using SGD optimizer and learning rate 0.01 with a precision value of 98.5%, recall value of 95.7%, f1-score value of 97.1%, and mAP value of 95.5%. In the future, we aim to expand the number of datasets and employ a broader range of hyperparameter values to optimize the classification precision and accuracy of the *Hiragana* Letter Detection system.**

***Keywords*—*Hiragana* characters; YOLOv8; Convolutional Neural Network (CNN); Python. mAP; recall; F1score; precision.**

Manuscript received 11 Jun. 2024; revised 26 Aug. 2024; accepted 25 Oct. 2024. Date of publication 30 Nov. 2024.

I. INTRODUCTION

In today's modern era, finding foreign languages online is easy. Various kinds of letters make some languages unique and have their characteristics. One is Japanese, which has many artworks such as anime, manga, and novels. Many are fond of Japanese art, not only Indonesian people but also the whole world, from students to workers, making Japan one of the places to study and work. Japanese has a sentence structure consisting of the letter's *Hiragana*, *Katakana*, and *Kanji* [1]. These three types of letters have their own writing rules. The letter hiragana is a phonetic symbol. Each letter is one mention of syllable *Hiragana* letters are formed from curved lines and strokes.

There are 46 characters of *Hiragana* letters; it is very challenging to research because the writing of *Hiragana* letters has many forms, and each human who writes it has its characteristics. Many people ignore the writing rules as long as the writing looks similar to the character, which makes the *Hiragana* letters written by each person varied and different [2]. As a result, it isn't easy to detect the writing for people new to *Hiragana* letters for the first time. Some research has been done to detect *Hiragana* handwriting. Research conducted by [2] using a combined CNN-SVM method to detect *Hiragana* and katakana handwriting. The dataset used was taken from respondents using a questionnaire. The results of this study using the system using only the CNN method have an accuracy of 87.82%, and the results obtained by the combined CNN and SVM method are 88.21%. A previous study by [3] conducted research using the CNN method to detect *Hiragana* letters. The dataset was taken from the internet and included as many as 1000 images. The study results achieved 95% accuracy with 950 images as training data and 50 as test data.

Research has been conducted using the YOLOv8 method by [4] using the YOLOv8 model to detect protective masks. The dataset is taken from Kaggle, which has 690 images using masks, 686 images without masks, and 703 images using masks incorrectly. The total dataset is 2079 images. The results of this study obtained several performances from YOLOv8, namely Precision 95%, Recall 95%, mAP (0.5) 96% and mAP (0.95) 93% using 200 epochs.

Based on earlier studies, this research will detect *Hiragana* letter objects using CNN with the YOLOv8 algorithm. YOLOv8 is the latest model of YOLO, designed to be fast, accurate, easy to use, and the right choice for detecting

Hiragana letter objects and analyzing the system's performance. YOLOv8 system performance includes Accuracy, Recall, Precision, F1-score, and Mean Average Precision (mAP).

II. Materials and Methods

A. Hiragana Characters

Hiragana is one of the basic letters of the kanji form. When it first began to develop, *Hiragana* was not immediately used by Japanese people because the Japanese aristocrats at the time refused to use *Hiragana* and only wanted to use kanji [27]. Historically, *Hiragana* was popularly used only by women and was also called *onnade* letters (women's letters). The *Hiragana* letters total 104 letters consisting of 46 basic letters, 25 *tentenmaru* letters, and 33 mixed letters. *Hiragana* letters have rules for writing letters because they considerably affect the shape of the letter itself [2].

B. Deep Learning

Deep learning, a machine learning type involving multiple layers of nonlinear data processing, is used forfeiture extraction, pattern recognition, and classification [5]. Deep learning is very good for supervised and unsupervised learning in various applications such as image recognition, voice, text classification, and so on [6]. Supervised learning is a model that learns to input images with labels, while unsupervised learning learns to input images without needing to label the data.

C. Convolutional Neural Network

NeoCognitron, the term used to describe the Convolutional Neural Network (CNN), was created by Kunihiko Fukushima, an expert at NHK Broadcasting Science Research Laboratories in Tokyo, Japan [7]. Convolutional neural network (CNN) is a Neural Network method developed and inspired by human nerves that can distinguish and recognize objects [8]. Convolutional Neural Network can be considered included in the Neural Network because it has a considerable depth that can be used as an implication in an image [9]. CNN is unique because it has a hidden layer that is only connected to a subset of neurons from the previous layer [30]. CNN is formed from neurons arranged to form a filter with a length and height with weight, bias, and activation, but overall, CNN is not much different from other deep learning algorithms [10].

The CNN architecture is divided into two main parts: the hidden and fully connected layers (MLP). The hidden layer converts input images into functions that contain numbers representing the image. In the fully connected layer, the feature map of the hidden layer, in the form of a matrix, is converted into one row so it can be linearly classified [23], [24]. The Convolutional Neural Network architecture is shown in Fig. 1.

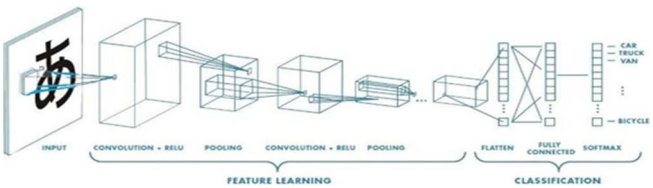

Fig. 1 Convolutional neural network layer model

D. Convolutional Network

The convolutional layer is the core part of a CNN, where the convolution operation occurs between the input image and the kernel (filter) that produces a new pixel. The convolutional layer technique helps clarify the input image pixels by replacing the image pixel values with values close to the image's original pixels. Using convolution does not change the dimension of the image. The convolutional layer has input consisting of length × height × depth. The convolution process is shown in Fig. 2; the input image is 5×5×1 with the kernel having a 3×3×1 matrix; because the stride value is 1 then, the kernel will shift nine times.

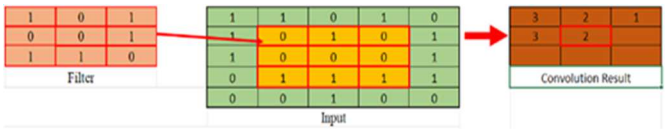

Fig. 2 Convolution matrix 5×5×1 input with 3×3×1 kernel

E. Pooling

Pooling, also called downsampling, reduces the matrix size of the input, which is helpful for a more effective model training process. There are two types: average pooling and max pooling [25]. Many people generally use max pooling because it selects the maximum value of the matrix region, while average pooling takes the average value [26]. The pooling process can be seen in Fig. 3.

Fig. 3 Max Pooling and Average Pooling

F. Stride

The stride is a parameter that determines how far the kernel (filter) moves to get information from the input image. If the stride value is 1, then the kernel (filter) moves by 1 pixel horizontally and then vertically. If the stride value is small, more accurate information is obtained from the input image, but it requires longer computation time when compared to using a larger stride value [11], [31]. The 1×1 stride process is shown in Fig. 4.

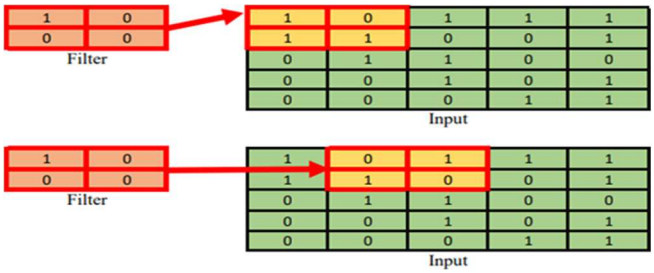

Fig. 4 Stride 1x1

G. Padding

Padding is a parameter added to each side of the input pixel that is 0. This step manipulates the output so that the resulting output does not lose information when convolution is running [12]. The addition of 1×1 padding is shown in Fig. 5.

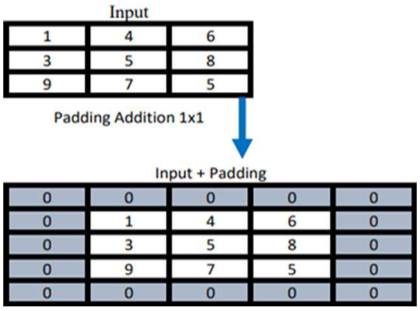

Fig. 5 Padding addition 1x1

H. Rectified Linear Unit (ReLU)

Rectified Liner Unit (ReLU) is a form of activation function widely used in deep learning. This activation function converts the negative pixel value in an image into 0 [13], which can be calculated using (1).

$$f(x) = \max(0, x) \tag{1}$$

I. Python

Guido Van Rossum is a Dutch computer programmer who created Python. Python was first released in 1991; the naming of this programming language was obtained from Monty Python's Flying Circus show because Guido Van Rossum was very fond of the show. Python can operate on various platforms: WindowsOS, MacOS, Linux and Raspberry Pi. Python is a versatile and user-friendly language, while other languages can be challenging to understand, with an emphasis on simple code [14], [21]. This makes Python very easy for beginners and those who have been learning programming languages for a long time. Python has a vast library to help programmers create sophisticated applications using simple source code [15].

J. YOLOv8

YOLO is an object detection algorithm [22], and YOLOv8 is the latest model of YOLO released in January 2023 and developed by Ultralytics, the developer of YOLOv5. Similar to YOLOv5, the architecture of YOLOv8 consists of a body and a head, which provides a new architecture with a better convolutional layer and an improved detector, which will become the first choice of many people for real-time object detection [16], [20]. YOLOV8 is still under development until this research is conducted, so other researchers may still make improvements, modify, or add new features.

K. YOLOv8 Architecture

YOLOv8 is a single-stage detector model that consists of two main parts: the backbone network and the head network. It uses a backbone network like YOLOv5 with some changes to the CSP layer, which is now called the C2f module. The C2f module helps improve detection accuracy. The YOLOv8 model uses a modified CSPDarknet53 architecture to become the backbone network. CSPDarknet53 has 53 convolution layers to extract essential features from the input image.

L. YOLOv8 Models

YOLOv8 offers five versions: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large) [16], [17]. The difference between all models is in the computation. The bigger the model, the longer the calculation. The specifications of each model are shown in Table 1.

TABLE I
YOLO V8 MODELS

Model	YOLO				
	v8n	v8s	v8m	v8l	v8x
Depth	0.33	0.33	0.67	1.00	1.00
Width	0.25	0.50	0.75	1.00	1.25
MaxChannels	1024	1024	768	512	512
Ratio	2.0	2.0	1.5	1.0	1.0
Layer	225	225	295	365	365
Parameter	3157200	11166560	25902640	43691520	68229648
Gradient	3157184	11166544	25902624	43691504	68229632
GFlops	8.9	28.8	79.3	165.7	258.5

M. Non-Maximum Suppression

Non-maximum suppression (NMS) is an object detection algorithm that reduces overlapping bounding boxes, resulting in varying confidence values [29]. This leads to incorrect placement of detected objects. This algorithm often creates several bounding boxes around the same object with different confidence values [16]. Non-maximum suppression works by taking the highest confidence value and eliminating the confidence value below it. The way non-maximum suppression works can be seen in Fig. 6.

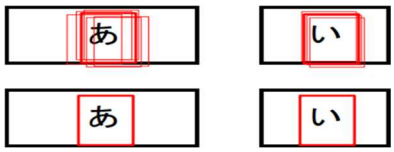

Fig. 6 Non-Maximum Suppression

N. Intersection Over Union

The object detection algorithm produces an output as a bounding box according to the system's prediction of the detected object in the input. To evaluate the overlapping bounding box values in the trained model, we can use the Intersection over Union (IoU) method. IoU is the ratio of the intersection area to the total area bounding box of the detection result and the ground truth [16][19]. Intersection over union can be seen in Fig. 7.

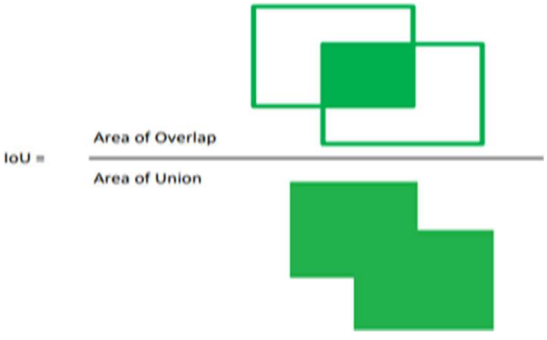

Fig. 7 Intersection Over Union

Mathematically, intersection over union can be calculated using (2).

$$\text{IoU} = \frac{GT \cap DR}{GR \cup DR} \quad (2)$$

where GT is Ground True, DR is Detection Result, and IoU is Intersection Over Union.

O. YOLOv8 Design

This final project aims to design a *Hiragana* letter detection system using a convolutional neural network with the YOLOv8 algorithm.

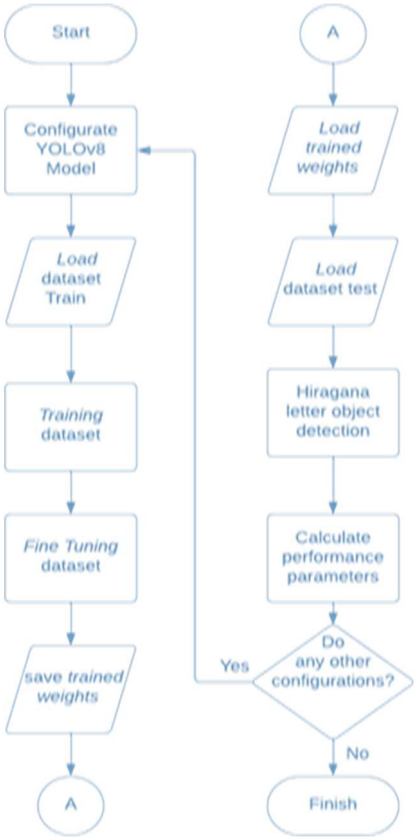

Fig. 8 YOLOv8 design

In the system that will be designed, as shown in Fig. 8, the detection will start from the configuration process of 5 types of YOLOv8, followed by the training process to train the model and fine-tuning, which aims to optimize the trained model to minimize overfitting. Then enter the model testing process, detect hiragana letters, and calculate performance parameters. Once the configuration has been completed, it is possible to test alternative hyperparameter values, including learning rate, batch size, channels, and max batches.

P. Performance Parameters

This step calculates the performance of the YOLOv8 model for detecting handwritten Hiragana letters. The performance parameters include Accuracy, Recall, Precision, F1-Score, and mAP.

1) Confusion Matrix: The confusion matrix, or error matrix, provides information about how the model's classification results compare with the actual results [28]. Itrix contains a table with four different combinations of model classification results and real results. The confusion matrix is shown in Figure 9.

		Actual Value: 1 (positive)	Actual Value: 0 (Negative)
Predicted Value	1 (Positive)	True Positive (TP)	False Positive (FP)
	0 (Negative)	False Negative (FN)	True Negative (TN)

Fig. 9 Confusion Matrix

The confusion matrix has 4 terms that represent the results of the classification process, namely true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

2) Accuracy: Accuracy is a parameter used to determine how accurate the system correctly detects objects. The accuracy can be calculated with equation (3).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \quad (3)$$

Where TP means for true positive, TN means for true negative, FP means for false positive, and FN stands for false negative.

3) Recall: Recall is the success of a model in re-detecting information. Recall can be calculated with equation (4).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

Where TP means for true positive and FP means for false negative.

4) F1-Score: F1-Score is the average comparison of precision and recall values. F1-Score has the highest value of 1 and the lowest of 0, if the F1-Score value is closer to 1, it shows that the system performance works well. F1-Score can be calculated with equation (6).

$$\text{F1-Score} = 2 \times \frac{Recall \times Precision}{Recall + Precision} \times 100\% \quad (6)$$

5) Mean Average Precision (mAP): The mean average precision is the average of precision values of all classes and measures how well the performance of weights from training data results. Before calculating mAP, adjusting the threshold on the IoU is necessary to validate the detected object. mAP can be calculated by equation (7).

$$\text{mAP} = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \quad (7)$$

III. Results and Discussion

This study aims to test and analyze the performance parameters of detecting handwritten hiragana letter objects using the YOLOv8 model. The dataset comprises 46 classes of hiragana letters from the internet by the author. For annotated authors using Roboflow[18]. The *Hiragana* letter dataset comprises 46 letters, with a division of 70% training data, 20% validation data, and 10% test data. The dataset comprises 4,600 letters, with 3,240 included in the training set, 950 in the validation set, and 410 in the test set. This test encompasses several performance parameters, including mAP, recall, f1-score, and precision. This test used several optimizers, such as SGD and Adam. The learning rates used are 0.01, 0.001, and 0.0001. This test is conducted to see how

optimal the mAP, precision, recall, and f1-score values are on each YOLOv8 model using an epoch value of 150.

A. Testing On YOLOv8n

Table 2 shows the performance when testing the YOLOv8n model for the SGD optimizer. All learning rates can run to completion without any obstacles. Still, at a learning rate of 0.0001, overfitting occurs, so the results are low because the smaller the learning rate, the faster the dataset becomes, burdening the computation.

TABLE II
TESTING ON YOLOV8N

Optimizer	Learning Rate	Precision	Recall	F1-Score	mAP
SGD	0.01	97.5%	96.2%	96.8%	94.7%
	0.001	97.4%	94.4%	95.9%	93.7%
	0.0001	56.2%	63.5%	59.6%	67.7%
Adam	0.01	94.7%	90.4%	92.5%	88.6%
	0.001	98.5%	95.8%	97.1%	94.8%
	0.0001	97.1%	95.4%	96.2%	93.8%

In contrast to Adam's optimizer, at learning rates 0.01 and 0.0001, overfitting occurs, which results in stopping computation at epoch 106 for learning rate 0.01 and epoch 120 for learning rate 0.0001 because the use of a significant learning rate will cause learning at the beginning to be faster before the rate is updated and when using a small learning rate will slow down learning when the data is trained. The best system for testing the YOLOv8n model is the Adam optimizer with a learning rate of 0.001, a Precision value of 98.5%, a Recall value of 95.8%, an F1-score value of 97.1%, a mAP value of 94.8%.

B. Testing On YOLOv8s

Table 3 shows the performance when testing the YOLOv8s model for all optimizers. Learning rates run smoothly, and the fit of the hyperparameters with this model is perfect because none of them occur overfitting when the simulation is run.

TABLE III
TESTING ON YOLOV8S

Optimizer	Learning Rate	Precision	Recall	F1-Score	mAP
SGD	0.01	97.8%	95.7%	96.7%	94.9%
	0.001	97.9%	94.4%	96.1%	94.8%
	0.0001	97.2%	94.5%	95.8%	93.7%
Adam	0.01	98.3%	95.4%	96.8%	94.5%
	0.001	98.4%	95.9%	97.1%	94.9%
	0.0001	97.9%	96.0%	96.9%	94.8%

The SGD optimizer with a learning rate of 0.01 and the Adam optimizer with a learning rate of 0.001 have the same mAP value of 94.9%, meaning that both hyperparameters get similar performance and fit this model. Still, for other values, the Adam optimizer is higher than the SGD optimizer. The best system from this model is the Adam optimizer, with a learning rate of 0.001 a precision value of 98.4%, a recall value of 95.9%, an f1-score value of 97.1% and a mAp value of 94.9%.

C. Testing On YOLOv8m

Table 4 shows the performance when testing the YOLOv8m model; the SGD optimizer experienced a decrease in value, and at a learning rate of 0.001, overfitting occurred, which only lasted at epoch 110. This is due to the model's incompatibility with channel depth and channel widening.

TABLE IV
TESTING ON YOLOV8M

Optimizer	Learning Rate	Precision	Recall	F1-Score	mAP
SGD	0.01	97.4%	96.4%	96.9%	95.2%
	0.001	97.4%	96.6%	97.0%	95.4%
	0.0001	96.9%	95.0%	95.9%	94.9%
Adam	0.01	96.3%	97.5%	96.9%	94.7%
	0.001	98.3%	96.4%	97.3%	95.1%
	0.0001	97.5%	95.5%	96.5%	95.0%

Adam optimizer has the best mAP value at a learning rate of 0.001 out of 95.1%; this learning rate is the best value of Adam optimizer. The best system of this model is the SGD optimizer with a learning rate of 0.01, a precision value of 97.4%, a recall value of 96.4%, an f1-score value of 96.9%, and a mAP value of 95.2%.

D. Testing On YOLOv8l

It can be seen in Table 5 performance when testing the YOLOv8l model. The SGD optimizer experiences overfitting at a learning rate of 0.001, which only lasts at epoch 147, and a learning rate of 0.0001 at epoch 138. This is due to a mismatch in the number of iterations, channel depth, and large channel widening. Adam's optimizer has no problem, and a learning rate of 0.001 is higher than other learning rates due to the hyperparameters' fit with channel depth and channel widening.

TABLE V
TESTING ON YOLOV8L

Optimizer	Learning Rate	Precision	Recall	F1-Score	mAP
SGD	0.01	98.5%	95.7%	97.1%	95.5%
	0.001	97.8%	95.3%	96.5%	94.8%
	0.0001	97.4%	95.9%	96.6%	94.6%
Adam	0.01	97.6%	95.8%	96.7%	95.0%
	0.001	97.2%	97.1%	97.1%	95.4%
	0.0001	97.6%	94.7%	96.1%	94.8%

The best system from this model is the SGD optimizer with a learning rate of 0.01, a precision value of 98.5%, a recall value of 95.7%, an f1-score value of 97.1%, and a mAP value of 95.5%.

E. Testing On YOLOv8x

Table 6 shows that when testing the YOLOv8x model, overfitting occurs in the SGD optimizer with a learning rate of 0.001, which survives at epoch 144, and the Adam optimizer with a learning rate of 0.0001, which survives at epoch 139. This is because the hyperparameter value used does not match the channel depth and channel widening that occurs.

TABLE VI
TESTING ON YOLOV8X

Optimizer	Learning Rate	Precision	Recall	F1-Score	mAP
SGD	0.01	98.1%	95.8%	96.9%	95.3%
	0.001	98.6%	95.2%	96.9%	95.3%
	0.0001	98.1%	94.9%	96.5%	95.4%
Adam	0.01	97.4%	96.2%	96.8%	94.5%
	0.001	98.5%	95.4%	96.9%	95.1%
	0.0001	97.9%	96.3%	97.1%	95.3%

The best system from this model is the SGD optimizer with a learning rate of 0.0001, which has a precision value of 98.1%, a recall value of 94.9%, an f1-score value of 96.5%, and a mAP value of 95.4%.

F. Comparison of Map Improvement in Each Model

Table 7 shows that the most optimal model is YOLOv8l with an SGD optimizer and learning rate of 0.01, with a mAP value of 95.5%. The mAP value decreases in the YOLOv8x model because the channel widening value and channel depth increase, resulting in overfitting during testing.

TABLE VII
COMPARISON OF MAP IMPROVEMENT IN EACH MODEL

Model	Optimizer	Learning Rate	mAP
YOLOv8n	Adam	0.001	94.8%
YOLOv8s	Adam	0.001	94.9%
YOLOv8m	SGD	0.01	95.2%
YOLOv8l	SGD	0.01	95.5%
YOLOv8x	SGD	0.0001	95.4%

YOLOv8n, YOLOv8s and YOLOv8m have an increase in the mAP value because the value of channel depth and channel widening is increasing for each model, causing more precise testing even though it causes an increasing computational burden as well. It can be seen in Table 4.6 that the Adam optimizer excels in the YOLOv8n and YOLOv8s models, which have a small channel depth and channel widening, but the SGD optimizer excels in the YOLOv8m, YOLOv8l, and YOLOv8x models which have a larger channel depth and channel widening.

G. YOLOv8 Model Test Results

The test results of this model test carried out on the YOLOv8l model with the SGD optimizer and learning rate of 0.01 show that the model selection is based on the test results for the most optimal mAP value.

TABLE VIII
TEST RESULTS OF ALL CLASS YOLOV8L MODELS

Class	Precision	Recall	F1-score	mAP
ALL	98.3%	96.4%	97.1%	95.5%
あ= A	88.8%	100%	94.2%	92.1%
い= I	96.3%	100%	99.6%	97.1%
う= U	99.1%	100%	99.8%	87.8%
え= E	95.2%	100%	89.6%	92.3%
お= O	100%	89.8%	93.8%	92.7%
か= KA	100%	90.6%	93.3%	96.7%
き= KI	100%	97.9%	98.1%	96.7%
く= KU	96.6%	92.6%	93.5%	92.4%
け= KE	100%	95.8%	91.8%	96.2%
こ= KO	100%	89.5%	94.0%	93.7%
さ= SA	96.7%	100%	99.1%	98.8%
し= SHI(SI)	99.1%	100%	99.6%	91.4%
す= SU	98.3%	100%	99.1%	98.7%
せ= SE	98.1%	100%	98.8%	99.0%
そ= SO	98.7%	100%	99.6%	94.3%
た= TA	98.6%	100%	99.1%	99.5%
ち= CHI(TI)	97.6%	100%	99.2%	95.1%
つ= TSU(TU)	96.7%	100%	99.6%	93.3%
て= TE	100%	95.3%	99.6%	97.6%
と= TO	99.6%	100%	99.7%	91.9%
な= NA	100%	95.6%	93.0%	97.7%
に= NI	100%	88.6%	93.9%	96.6%
ぬ= NU	98.7%	100%	99.5%	96.0%
ね= NE	100%	95.3%	96.3%	99.2%
の= NO	99.1%	100%	97.4%	97.9%
は= HA	84.4%	100%	99.0%	98.7%
ひ= HI	99%	100%	99.7%	95.1%
ふ= FU(HU)	97.8%	100%	98.0%	93.6%
へ= HE	99.5%	100%	99.8%	90.3%
ほ= HO	100%	95.2%	99.9%	95.6%
ま= MA	100%	85.5%	95.0%	97.6%
み= MI	100%	94.9%	93.3%	98.1%
む= MU	100%	90.5%	93.2%	92.6%
め= ME	100%	89.6%	94.2%	98.7%
も= MO	100%	84.1%	91.4%	98.8%
や= YA	99%	100%	99.8%	92.2%
ゆ= YU	98.4%	100%	99.1%	93.2%
よ= YO	98.5%	100%	99.5%	89.9%
ら= RA	100%	98%	96.5%	97.8%
り= RI	98.4%	100%	98.2%	95.9%
る= RU	96.3%	92.3%	95.0%	93.9%
れ= RE	100%	84.3%	96.4%	98.7%
ろ= RO	99.4%	100%	99.7%	99.5%
わ= WA	96.3%	100%	98.4%	95.8%
を= WO	99.1%	100%	99.7%	95.1%
ん= N	96.5%	88.6%	91.3%	95.1%

The highest mAP value is in the class "TA" and "RO" of 99.5% indicating that the model used is very suitable for the test, the learning carried out by YOLOv8 is very good for these two letters. There are some letters with low mAP values with the lowest value of 87.8% found in class "U", this is because the dataset used is handwritten and also some hiragana letters are unique because they have their own rules in writing, which causes the letter to have many forms.

H. Comparison of Results

In previous research using 2 models, namely the CNN model and the CNN-SVM combined model, the mAP value was 87.82% and 88.21% using the Handwritten dataset. In this Final Project the author uses datasets collected from various sources with more variety.

TABLE IX
COMPARISON OF PREVIOUS RESEARCH RESULTS

Model	Dataset	mAP
CNN	Handwritten Hiragana	87.82%
CNN-SVM	Handwritten Hiragana	88.21%
YOLOv8	Handwritten Hiragana	95.50%

This research uses the latest model from YOLO, namely YOLOv8, and gets the most optimal results in YOLOv8l using the SGD optimizer with a learning rate of 0.01 and an mAP value of 95.5%. In this test, the authors succeeded in increasing the mAP from previous studies, namely 87.82% and 88.21%, to 95.5% by using the YOLOv8l model using similar datasets to previous studies.

IV. CONCLUSION

This study discussed the analysis of the classification of handwritten Hiragana letters, focusing on the efficacy of different optimization techniques, namely SGD and Adam, and the use of three different learning rate values, 0.01, 0.001, and 0.0001. The analysis is conducted by comparing previous

research utilizing a CNN architecture, combined with a CNN-SVM, and the proposed YOLOv8 model. Upon testing the system, it can be concluded that the YOLOv8l model is the most effective model when tested using the SGD optimizer with a learning rate of 0.01, achieving a precision value of 98.5%, a recall value of 95.7%, an F1 score value of 97.1% and a mAP value of 95.5%. In this study, the author demonstrated an improvement in the mAP value from the previous research conducted using the CNN model and the combined CNN-SVM model, with values of 87.82% and 88.21%, respectively, to a mAP value of 95.5%, achieved using the YOLOv8l model. The objective of this research is to assist individuals in the accurate and precise identification of hiragana letters. In the future, we aim to expand the number of datasets and employ a broader range of hyperparameter values to optimize the classification precision and accuracy of the Hiragana Letter Detection system.

REFERENCES

[1] A. H. Mawaddah, C. Atika Sari, D. R. Ignatius Moses Setiadi, and E. Hari Rachmawanto, "Handwriting Recognition of Hiragana Characters using Convolutional Neural Network," 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), pp. 79–82, Sep. 2020, doi:10.1109/isemantic50169.2020.9234211.

[2] N. E. W. Nugroho and A. Harjoko, "Transliteration of Hiragana and Katakana Handwritten Characters Using CNN-SVM," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 15, no. 3, p. 221, Jul. 2021, doi: 10.22146/ijccs.66062.

[3] C. Umam, A. D. Krismawan, and R. R. Ali, "CNN for Image Identification of Hiragana Based on Pattern Recognition using CNN," Journal of Applied Intelligent System, vol. 6, no. 2, pp. 62–71, Dec. 2021, doi: 10.33633/jais.v6i2.4586.

[4] Tamang, S., Sen, B., Pradhan, A., Sharma, K., & Singh, V. K. (2023). Exploring YOLOv8 Object Detection for Accurate Face Mask Classification. *Ijisae, 2023*(2), 892–897.

[5] M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," Computers, vol. 12, no. 5, p. 91, Apr. 2023, doi:10.3390/computers12050091.

[6] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," SN Computer Science, vol. 2, no. 6, Aug. 2021, doi: 10.1007/s42979-021-00815-1.

[7] K. Kaplan, Y. Kaya, M. Kuncan, and H. M. Ertunç, "Brain tumor classification using modified local binary patterns (LBP) feature extraction methods," Medical Hypotheses, vol. 139, p. 109696, Jun. 2020, doi: 10.1016/j.mehy.2020.109696.

[8] A. Anton, N. F. Nissa, A. Janiati, N. Cahya, and P. Astuti, "Application of Deep Learning Using Convolutional Neural Network (CNN) Method For Women's Skin Classification," Scientific Journal of Informatics, vol. 8, no. 1, pp. 144–153, May 2021, doi:10.15294/sji.v8i1.26888.

[9] Y. A. Suwitono and F. J. Kaunang, "Implementasi Algoritma Convolutional Neural Network (CNN) Untuk Klasifikasi Daun Dengan Metode Data Mining SEMMA Menggunakan Keras," Jurnal Komtika (Komputasi dan Informatika), vol. 6, no. 2, pp. 109–121, Nov. 2022, doi: 10.31603/komtika.v6i2.8054.

[10] M. S. Elsayed, H. Z. Jahromi, M. M. Nazir and A. D. Jurcut, "The role of CNN for intrusion detection systems: An improved CNN learning approach for SDNs", Proc. Int. Conf. Future Access Enablers Ubiquitous Intell. Infrastructures, pp. 91-104, 2021.

[11] F. Alantali, Y. Halawani, B. Mohammad, and M. Al-Qutayri, "SLID: Exploiting Spatial Locality in Input Data as a Computational Reuse Method for Efficient CNN," IEEE Access, vol. 9, pp. 57179–57187, 2021, doi: 10.1109/access.2021.3071409.

[12] A. R. Fauzy. "Implementasi Metode Region Convolutional Neural Network Dalam Mendiagnosa Anomali Pneumonia Pada Foto Thorax", Elibrary Unikom. PhD diss., Universitas Komputer Indonesia, 2019.

[13] R. B. J. Simanjuntak, Y. Fu'adah, R. Magdalena, S. Saidah, A. B. Wiratama, and I. D. S. Ubaidah, "Cataract Classification Based on Fundus Images Using Convolutional Neural Network," JOIV : International Journal on Informatics Visualization, vol. 6, no. 1, p. 33, Mar. 2022, doi: 10.30630/joiv.6.1.856.

[14] V. Cutting, & N. Stephen. "A Review on Using Python as A Preferred Programming Language for Beginners". *International Research Journal of Engineering and Technology,* 2021, vol. 8, no. 8, pp.4258–4263.

[15] J. Terven and D. Cordova-Esparza. (2023). A comprehensive review of YOLO architectures in computer vision: from YOLOV1 to YOLOV8 and YOLO-NAS," *arXiv (Cornell University)*, doi:10.48550/arxiv.2304.00501.

[16] E. Panja, H. Hendry, and C. Dewi, "YOLOv8 Analysis for Vehicle Classification Under Various Image Conditions," Scientific Journal of Informatics, vol. 11, no. 1, pp. 127–138, Feb. 2024, doi:10.15294/sji.v11i1.49038.

[17] Q. Xu, Y. Wei, J. Gao, H. Yao, and Q. Liu, "ICAPD Framework and simAM-YOLOv8n for Student Cognitive Engagement Detection in Classroom," IEEE Access, vol. 11, pp. 136063–136076, 2023, doi:10.1109/access.2023.3337435.

[18] J. M. Kimeu, M. Kisangiri, H. Mbelwa, and J. Leo, "Deep learning-based mobile application for the enhancement of pneumonia medical imaging analysis: A case-study of West-Meru Hospital," Informatics in Medicine Unlocked, vol. 50, p. 101582, 2024, doi:10.1016/j.imu.2024.101582.

[19] U. Ali, M. A. Ismail, R. A. Ariyaluran Habeeb, and S. R. Ali Shah, "Performance Evaluation of YOLO Models in Plant Disease Detection," Journal of Informatics and Web Engineering, vol. 3, no. 2, pp. 199–211, Jun. 2024, doi: 10.33093/jiwe.2024.3.2.15.

[20] P.-W. Chin, K.-W. Ng, and N. Palanichamy, "Plant Disease Detection and Classification Using Deep Learning Methods: A Comparison Study," Journal of Informatics and Web Engineering, vol. 3, no. 1, pp. 155–168, Feb. 2024, doi: 10.33093/jiwe.2024.3.1.10.

[21] K. Chromiński, L. Benko, Z. J. Hernández-Figueroa, J. D. González-Domínguez, and J. C. Rodríguez-del-Pino, "Python Fundamentals," Nov. 2021, doi: 10.17846/fpvai-2021-14.

[22] M. S. Z. Ahmad, N. A. Ab. Aziz, and A. K. Ghazali, "Development of Automated Attendance System Using Pretrained Deep Learning Models," International Journal on Robotics, Automation and Sciences, vol. 6, no. 1, pp. 6–12, Apr. 2024, doi: 10.33093/ijoras.2024.6.1.2.

[23] P. G. Solanellas. "Designing Convolutional Neural Networks for Classification of Metastatic Tissue in the Lymph Nodes". Bachelor's thesis, Universitat Politècnica de Catalunya, 2022.

[24] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely Automated CNN Architecture Design Based on Blocks," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 4, pp. 1242–1254, Apr. 2020, doi: 10.1109/tnnls.2019.2919608.

[25] A. Wisnu, and H. Solichul, "Multichannel convolutional neural network model to improve compound emotional text classification performance," IAENG International Journal of Computer Science, vol. 50, no. 3, pp. 866–874, 2023.

[26] A. Zafar et al., "A Comparison of Pooling Methods for Convolutional Neural Networks," Applied Sciences, vol. 12, no. 17, p. 8643, Aug. 2022, doi: 10.3390/app12178643.

[27] H. Kunert, "Contemporary Loanwords inhiragana: An Analysis of Typical Traits and Contexts," Japanese Studies, vol. 40, no. 1, pp. 21–40, Jan. 2020, doi: 10.1080/10371397.2019.1682935.

[28] D. Krstinić, M. Braović, L. Šerić, and D. Božić-Štulić, "Multi-label Classifier Performance Evaluation with Confusion Matrix," Computer Science & Information Technology, Jun. 2020, doi:10.5121/csit.2020.100801.

[29] S. Jiang, T. Xu, J. Li, B. Huang, J. Guo, and Z. Bian, "IdentifyNet for Non-Maximum Suppression," IEEE Access, vol. 7, pp. 148245–148253, 2019, doi: 10.1109/access.2019.2944671.

[30] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification," IEEE Transactions on Cybernetics, vol. 50, no. 9, pp. 3840–3854, Sep. 2020, doi: 10.1109/tcyb.2020.2983860.

[31] M. Helmy, T. T. Truong, E. Jul, and P. Ferreira, "Deep learning and computer vision techniques for microcirculation analysis: A review," Patterns, vol. 4, no. 1, p. 100641, Jan. 2023, doi:10.1016/j.patter.2022.100641.