e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

OPTIMIZER COMPARISON IN CONVOLUTIONAL NEURAL NETWORK FOR REAL TIME FACE RECOGNITION

Elbert[1*], Meirista Wulandari[2], Joni Fat[3]
[1-3] Universitas Tarumanagara
elbert.525210005@stu.untar.ac.id; meiristaw@ft.untar.ac.id; jonif@ft.untar.ac.id

*Correspondence: jonif@ft.untar.ac.id

***Abstract** – Face recognition is one of the computer vision technologies that's used in many industries. Face recognition always used in various sector that require the verification of an individual identity. There are many ways that can be used to develop face recognition, one of them is convolutional neural network. Convolutional neural network (CNN) is a deep learning neural network that is created specifically to process and analyze visual data, such as images and videos. CNN have the ability to learn many features from visual data, making them highly effective for tasks like face recognition. There are many factors that can affect CNN performance including the optimizers that are used in the neural network. Optimizers are the algorithm that adjust weights of the neural network to minimize error between the predicted output and actual target. This study used 10 different subjects for face recognition. In this study, the CNN model uses a training algorithm called backpropagation then will compare 3 different types of optimizers. The optimizers that used in this study are Adaptive Momentum (Adam), Root Mean Square Propagation (RMSProp), and Stochastic Gradient Descent (SGD). The results of the comparison will be shown in the form of performance metrics including the confusion matrix. Based on the comparison, CNN model with SGD optimizer is proved to have the best performance includes correct classification rate (CCR), precision, recall and F1-score. CNN model with SGD optimizers has the highest CCR, precision, recall, and F1-score of 97.07%, 97.13%, 97.06%, and 97.06% respectively.*

Keywords: Face Recognition; Convolutional Neural Network; Adam; SGD; RMSProp

I. INTRODUCTION

Technological advances are becoming more extensive and rapid as time goes by. Face recognition is one of various types of technology that can help human in many fields. Face Recognition is a technique that uses human facial biometrics to identify a person (Cheng et al., 2021). Face recognition can be applied in various fields to verify a person's identity, such as border inspection, building access control, criminal identification, and user identification for a device. Face recognition can be designed using various methods, one of the methods is convolutional neural network.

Convolutional neural network (CNN) is a type of algorithm in *deep learning* for processing various data such as image data (Dacipta & Putra, 2022). CNN is generally used to recognize objects in images. CNN is often used in fields that require image processing such as tumor cells, fingerprints, and flower species (Bhatt et al., 2023). CNN has also been applied in real-time such as automated vehicles.

There are some research about CNN for face recognition that have been carried out. In a research article entitled “Smart Recognition of Real Time Face Using Convolutional Neural Network (CNN) Technique” by Alagarsamy et al, there is research on face recognition using a CNN model called FaceNet (Alagarsamy et al., 2020). According to research conducted by Alhanaee et al with entitled “ Face Recognition

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

Smart Attendance System using Deep Transfer Learning", there is a research about creating an attendance system based on face recognition using the transfer learning method on the AlexNet, GoogleNet, and SqueezeNet CNN models (Alhanaee et al., 2021).

The CNN model carries out training to achieve accurate results. There are various training methods that can be used in CNN, one of them is backpropagation. Backpropagation is a training method that allows a neural network to optimize the output values produced through error learning and changing the neural network weight values (Solihat et al., 2024). While using backpropagation algorithm, optimization is carried out using an optimization algorithm or optimizer.

Optimizer is an algorithm that used to change a parameter in a neural network (Kurniawan et al., 2023). This algorithm can change the weight values and learning rate in a neural network. This aims to minimize errors during the model training process. Optimizer is also an important parameter in the training process because it used to optimizer the parameter value for a neural network so that it can predict data accurately (Solihat et al., 2024).

There are various types of optimizers that can be used so they can produce varying results. Therefore, studies on comparative analysis of optimizers are important to determine the most effective optimizer, especially in the case of face recognition (Taqiyuddin et al., 2023). The aim of this research is to compare and evaluate different optimizers to produce an effective and accurate CNN model for face recognition.

The optimizers used in this research are Adaptive Momentum (Adam), Stochastic Gradient Descent (SGD), and Root Mean Square Propagation (RMSProp). These 3 optimizers are used because each optimization method is always used to carry out training in CNN. Each optimizers has a different approach in optimizing and updating CNN weight.

Adam calculates adaptive gradients as well as momentum gradients to optimize training efficiency. SGD computes the gradient of the loss function against each batch of training data and performs an inverse step against the gradient to update the model weights. RMSProp adapts the learning rate for each parameter based on the gradient change in the last iteration (Nurdin et al., 2024). By comparing the optimizers, people can make decisions that increase model efficiency and dependability, leading to impressive achievements across a range of domains (Amulya et al., 2024). The parameters used to compare CNN models with different optimizers are correct classification rate (CCR), precision, recall, F1-score, and confusion matrix.

II. METHODS

This research was conducted using certain hardware and software specifications. Ten face data participants which are captured by Virtual Graphic Array (VGA) camera are used for real-time face identification under certain height, distance, and positional constraints. The hardware specifications used in this research is a laptop with intel core i7 processor and 8 GB RAM. The software specifications used in this research are visual studio code version 1.96.0 and python version 3.9.11.

The system applies a single CNN architecture that has been developed specifically for this particular scenario. The CNN architecture is created using different optimizer which are Adam, SGD, and RMSProp. Then the CNN used to identify image data of the participant to obtain the performance value. The evaluation of the CNN is obtained using some criteria, such as picture size, accuracy, precision, recall, F1-score, and confusion matrix. These restrictions were designed to maintain the focus on the potential and effectiveness of the proposed model in a certain configuration, providing the benchmark for future improvements in various of circumstances.

The image data are processed with a processing module through 2 processes. They are the face detection process and the face identification

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

process. The face detection process involves detecting faces in images that was capture by image capturing modules. The face detection is carried out using haar cascade classifier. The face identification process includes identifying faces from images that was received from image capturing module. The face identification process is carried out using a CNN model that was created. The results of the processing module are displayed by using the information display module. The results are output information includes the names and faces that was identified. The block diagram for comparative analysis of optimizers in convolutional neural networks for face recognition is shown in Figure 1.

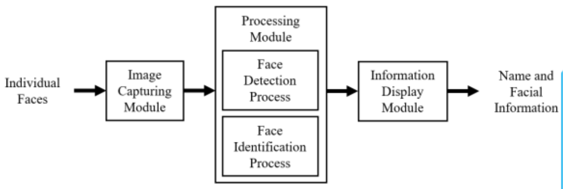
Individual Faces
Image Capturing Module
Processing Module
Face Detection Process
Face Identification Process
Information Display Module
Name and Facial Information

Figure 1. Block Diagram of Comparative Analysis of Optimizers in CNN for Face Recognition

2.1. Individual Face

An individual's face is one of the distinctive and unique aspects to determine a person's identity (Hanafie et al., 2023). In this study, individual face is used as an input for face recognition. Images of individual faces are captured by using an image capturing module. Examples of individual face images is shown in Figure 2.

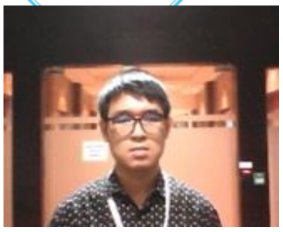

Figure 2. Individual Face

2.2. Image Capturing Module

The image capturing module is a component that used to take pictures of individual face. Images are taken in Red, Green, Blue (RGB) color format according to the specifications of the camera used. The image capture module used is a VGA camera. The image capturing module that used in this study is shown in Figure 3.

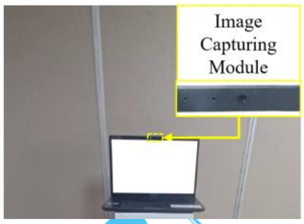

Figure 3. Image Capturing Module

2.3. Processing Module

The captured image is used as *input* into the processing module. The processing module is a component that used to process image data. The processing module used is a processor. The processing module processes the received image through 2 processes. They are the face detection process and the face identification process. The processing module that used in this study is shown in Figure 4.

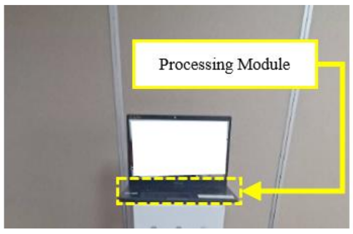

Figure 4. Processing Module

2.3.1 Face Detection Process

The processing module begins with the face detection process. The face detection process is the process of finding and detecting faces in the received image. The face detection process is carried out using the Haar Cascade Classifier. After the face image is detected, the resizing and reshaping process is carried out in order to adjust the size and shape of the facial image needed for the face identification process (Yulina, 2021).

2.3.1.1 Haar Cascade Classifier

The process of detecting face in images is carried out by using the haar cascade classifier algorithm. Haar cascade classifier is an

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

algorithm that used to detect objects by using a cascade function. This algorithm extracting features from images using a filter called haar features (Yulina, 2021). This filter produces various types of features such as edge features, line features, and four-rectangle features based on the part that was examined by the filter. Example of the feature that produced by haar features is shown in Figure 5

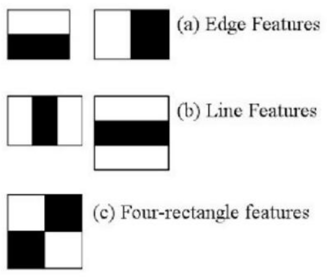

Figure 5. Example of The Features that Produced by Haar Feature (Yulina, 2021)

The image color format was changed from Red, Green, Blue (RGB) to grayscale so that the image can be used as input for the haar cascade classifier. This because the haar cascade classifier can only detect facial features in grayscale color format. In images, the haar cascade classifier can detect objects such as human faces and eyes. Then cropping is done on the detected face area in the input image. The detection result by using the haar cascade classifier is an image containing the face that was detected and cropped from the input image.

The detected face image is resized through a resizing process. Resizing is the process of changing the size of an image. The face image is changed according to the input size required by the CNN model to identify the face. Images that have been resized are further processed through a reshaping process. Reshaping involves changing the dimensions or shape of the data. The data must be reshaped into suitable input for a CNN model.

2.3.2 Face Identification Process

After the face detection process was done, the face image is identified using a convolutional neural network (CNN) model. CNN is a machine learning method developed from multi layer perceptron (MLP) which is designed to process 2 dimensional data (Lesmana et al., 2022). CNN relies on sliding windows or filters. CNN also adopt the principle of weight sharing. This means that every filter that has been shifted on the image will create a new feature on the image (Putra et al., 2023). The CNN model that used to identify the face image has been trained, validated and tested using a dataset consisting of various facial images with the aim of carrying out the identification process accurately. The image results from the face detection process are used as input in the CNN model. The CNN model identifies faces based on these images. There are two stages that must be done before CNN can be used to identifying face image. They are facial data preparation and training convolutional neural network (CNN) model.

2.3.2.1 Facial Data Preparation

Facial data preparation is required to train, validate, and test CNN models. The data that used in this study is facial data in image form. After the facial data is prepared, the facial images in each data are detected using the haar cascade classifier algorithm. After that, the detected facial images are increased through augmentation. Augmentation is the process of generating new data by using transformations on the original data (Anu et al., 2023).
Images that have undergone an augmentation process are given a label through a labeling process. Labeling is the process of giving a label or name to an object (Backar et al., 2023). Label encoding is carried out after getting the label. Label encoding is the process of changing the form of a label into a numerical representation (Hasyani et al., 2023). This process is carried out because the CNN model cannot process data in the form of categorical labels.

Encoding process is carried out using the one-hot encoding technique. One-hot encoding is a technique that convert the data labels into binary form (Anu et al., 2023). After that, the dataset is processed through a reshaping process. This aims to change the shape of the dataset into an input form that suits the CNN model. The reshaped dataset is divided into 3 parts. They are the train data, validation data,

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

and test data. These 3 pieces of data are used to train, validate, and test the CNN model.

2.3.2.2 Convolutional Neural Network

The face identification process is carried out using the CNN model. A CNN model needs to created based on an architecture before it can be trained, validated, tested, and identify individual faces. The CNN architecture consists of convolution layers, pooling layers, fully connected layers, and activation functions (Purwono et al., 2022). These layers are used to form the CNN architecture. An example of network architecture in a CNN is shown in Figure 6.

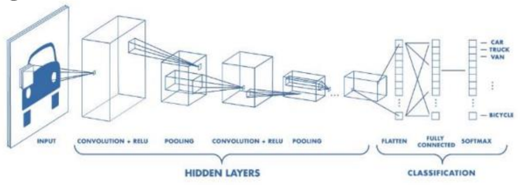

Figure 6. An Example of Architecture in A CNN (Purwono et al., 2022)

Convolutional layers are layers that use kernels filter to calculate convolutions and extract features from input images. Pooling layer is a layer that reduces the number of parameters and computational load by creating a down-sampling representation or resizing. Flatten layer is a layer that converts a multidimensional array into a vector for a fully connected layer. A fully connected layer is a layer that transforms the results of the previous pooling or convolutional layer into a vector. Activation function is a certain calculation function that is used to determine the output of layers and neural networks.

The CNN was designed using the Rectified Linear Unit (ReLU) and softmax activation function. The ReLU activation function is an activation function that makes all pixel values less than 0 in the image become 0. The equation used for the ReLU activation function is shown in **Error! Reference source not found.**.

$$f(x) = \max(0,x) \qquad (1)$$

The softmax activation function is an activation function that used for classifying more than 2 classes (Ilahiyah S & Nilogiri A, 2018). Activation function softmax allows calculation of probabilities for all classes. Equation for activation function softmax is shown in **Error! Reference source not found.**.

$$f_j(z)=\frac{e^{zj}}{\Sigma_k e^{zk}} \qquad (2)$$

All

Table 1. Architecture CNN Model

Layer	Layer Configuration
Convolution	32 filter, 3×3 kernel, and ReLU
Convolution	32 filter, 3×3 kernel, and ReLU
Max-Pooling	2×2 kernel
Convolution	64 filter, 3×3 kernel, and ReLU
Convolution	64 filter, 3×3 kernel, and ReLU
Max-Pooling	2×2 kernel
Convolution	128
Convolution	128
Max-Pooling	2×2 kernel
Flatten	8192 Neuron
Fully	512 Neuron, and ReLU
Output Layer	10 Classes, and Softmax

After

While

2.3.2.2.1 Adaptive Momentum (Adam)

Adam is an optimizer algorithm specifically designed for training a deep neural network. Adam uses bias value at 2 different moments to improve the weights/parameters in the CNN. Adam calculates the adaptive learning rate for each parameter that used in the training process (Iskandar Zulkarnain Maulana Putra et al., 2022). Adam uses exponentially averaged gradient with equation **Error! Reference source not found.**). Apart from that, Adam uses exponential mean of the squared gradient that involves parameter which is based on equation **Error! Reference source not found.**). The weight difference is calculated using learning rate and gradient based on equation $\Delta\omega$⍰t⍰ $= -\eta\frac{x_t}{\sqrt{y_t+\epsilon}} * g_t$ (*4*). The latest weight is calculated with equation ωt+1 = $\omega_t + \Delta\omega_t$ (*5*).

$$x_t = \delta_1 * x_{t-1} - (1 - \delta_1) * g_t \qquad (2)$$

$$y_t = \delta_2 * y_{t-1} - (1 - \delta_2) * g_t^2 \qquad (3)$$

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

$$\Delta\omega_t = -\eta \frac{x_t}{\sqrt{y_t+\in}} * g_t \quad (4)$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t \quad (5)$$

Equation Source: (Iskandar Zulkarnain Maulana Putra et al., 2022)

Based on the equation above, (η) represents the learning rate (Iskandar Zulkarnain Maulana Putra et al., 2022). The gradient at time (t) along (ω_j) is (g_t). The gradient along (ω_j) is averaged exponentially as (x_t). The exponential mean of the squared gradients along (ω_j) is represented by (y_t). Parameters are represented by (δ_1) and (δ_2). δ_1 is an exponential parameter to calculate the average momentum. δ_2 is an exponential parameter for calculating the mean squared gradient.

2.3.2.2.2

RMSProp is an optimization algorithm that have functions to reduce noise in a neural network by preventing errors from spreading throughout the entire neural network (Iskandar Zulkarnain Maulana Putra et al., 2022). RMSProp uses partial gradient descent averaging to adjust the step size of each weight. RMSProp adaptively adjusts the learning rate of each weight during training. RMSProp used the exponential average of the squared gradient that can be calculated using equation (6). Apart from that, RMSProp calculate the weight difference with equation $\Delta\omega_t = -\frac{\eta}{\sqrt{y_t+\in}} * g_t$ (7) which involves learning rate and gradient. The updated weight is calculated using equation $\omega t+1 = \omega_t + \Delta\omega_t$ (8).

$$v_t = \delta * v_{t-1} - (1 - \delta) * g_t^2 \quad (6)$$

$$\Delta\omega_t = -\frac{\eta}{\sqrt{y_t+\in}} * g_t \quad (7)$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t \quad (8)$$

Equation Source: (Iskandar Zulkarnain Maulana Putra et al., 2022)

Based on the equation above, RMSProp uses the exponential average of the squared gradient (v_t) and learning rate (η) (Iskandar Zulkarnain Maulana Putra et al., 2022). (g_t) is the gradient along (ω_j) at time (t). When a new training data is received, the neural network's weight values are changed during the training process. This happens by changing the value stored for each neuron.

2.3.2.2.3 Stochastic Gradient Descent (SGD)

SGD is an optimization algorithm that uses input to produce precise output (Alzubaidi et al., 2021). SGD is generally used for deep learning. SGD uses linear regression for optimize the parameter in a neural network. SGD used initial weight, learning rate, and error function that calculated the new weight value using equation (9). The initial weight is calculated based on equation (10). The error value is calculated using equation (11).

$$W = \omega - \eta \nabla Q_i(\omega) \quad (9)$$

$$W \leftarrow \eta \nabla Q(\omega) \quad (10)$$

$$Q(\omega) = \ln \sum i\, Q_i(\omega) \Rightarrow \nabla Q(\omega) = \ln \sum i\, \nabla Q_i(\omega) \quad (11)$$

Based on the equation above, the initial weight is represented by (ω) (Alzubaidi et al., 2021). (η) is the learning rate which determines how often the weight are changed. Q_i is the data that is being observed. In general, Q is an error function. SGD minimizes Q in order to produce the best weight value (W).

2.3. Information Display Module

The information display module receives facial image identification results from the processing module. The information display module shows the result form processing module including information about name and face that was identified. The information display module used in this study is a Liquid Crystal Display (LCD). The information display module is shown in Figure 7. Information Display Module

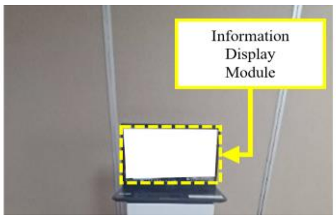

Figure 7. Information Display Module

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

2.4. Performance Metrics

Performance metrics are parameters used to determine the performance of machine learning or deep learning algorithms. In this study, performance metrics that will be shown include the correct classification rate (CCR) and confusion matrix. Confusion matrix is a table that states the correct and wrong classification based on the test data. Correct classification rate is the number of cases classified into the correct category based on the model prediction results.

III. RESULTS AND DISCUSSION

The study was conducted with 10 different subjects with 1000 data for each subject. The data that used is splitted into train data and test data with ratio 70%:30%, respectively. The results have been obtained were the performance metrics including CCR, precision, recall, F1-score, and confusion matrix. The CCR values, precision, recall, and F1-score was generated based on the CNN model performance. There are various CCR value, precision, recall, F1-score that was obtained with different optimizers such as Adam, RMSProp, and SGD. Those are widely used in CNN training and have various techniques for weight optimization, they were selected as the three optimizers for this research. Adaptive Moment Estimation, or Adam, is preferred because it employs adaptive learning rates for each parameter. Adam is particularly appropriate to solve issues involving noisy datasets and sparse gradients. RMSprop (Root Mean Square Propagation) is particularly effective for non-stationary objectives, as it divides the learning rate by a running average of recent gradient magnitudes. In particular, SGD (Stochastic Gradient Descent) offers a more conventional and uncomplicated method by gradually updating weights according to the gradient of the loss function. SGD is well-known for its ease of use and effectiveness. It performs particularly well when integrated with parameters such as momentum or learning rate schedules, which provide improved generalization and adaptability, particularly in larger datasets. These optimizers were selected as a result of an accurate evaluation of several optimization techniques to evaluate their effects on CNN training efficiency. The CCR, precision, recall, and F1-score values that obtained using different optimizers are shown in **Error! Reference source not found.**, Table 3, Tabel 4, and Table 5, respectively.

Table 2. Correct Classification Rate Value with Adam, RMSProp, and SGD Optimizer

Optimizer	Correct Classification Rate
Adam	97.00%
RMSProp	96.16%
SGD	97.07%

Table 3. Precision Value with Adam, RMSProp, and SGD Optimizer

Optimizer	Precision
Adam	97.08%
RMSProp	96.28%
SGD	97.13%

Table 4. Recall Value with Adam, RMSProp, and SGD Optimizer

Optimizer	Recall
Adam	97.00%
RMSProp	96.16%
SGD	97.06%

Table 5. F1-Score Value with Adam, RMSProp, and SGD Optimizer

Optimizer	F1-Score
Adam	97.00%
RMSProp	96.18%
SGD	97.06%

The confusion matrix displays the number of correctly and incorrectly identified images for each subject. The correctly identified images can be seen diagonally in the confusion matrix. Confusion matrix for CNN models with 3 different optimizer is shown in
Figure *8*.

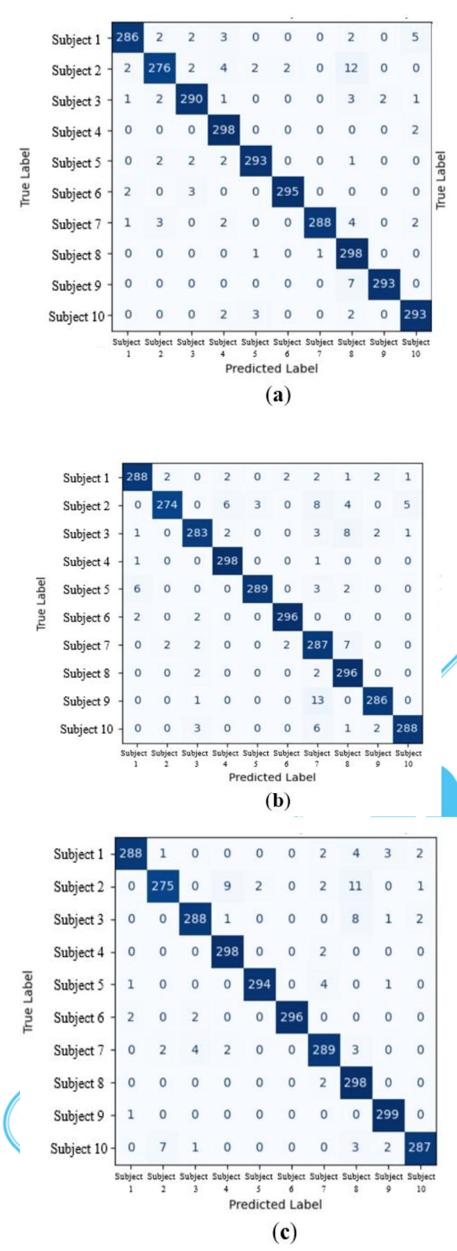

Figure 8. (**a**) Confusion Matrix CNN Model using Adam; (**b**) Confusion Matrix CNN Model using RMSProp; (**c**) Confusion Matrix CNN Model using SGD

Based on the confusion matrix that has been obtained, there are several facial images that are identified correctly. Example of an image that the face has been successfully identified is shown in Figure 9.

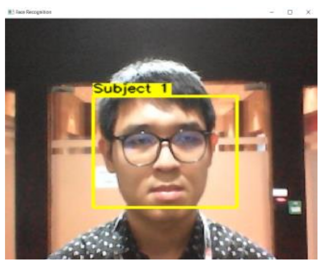

Figure 9. Example of a Successfully Identified Face Image

IV. CONCLUSION

CNN is a deep learning algorithm that performs face recognition. Performance of CNN is influenced by the optimizer that being used. There are many optimizers including Adam, SGD, and RMSProp that can be used in CNN. Those three optimizers could be compared to obtain the best optimizer in the CNN model. Based on research that has been carried out, the CNN model using the SGD optimizer has the highest CCR value of 97.07%, precision value of 97.13%, recall value of 97.06%, and F1-score value of 97.06%. CNN model with SGD optimizer has the best CCR, precision, recall, and F1-score value for real time face recognition. The CNN model using SGD optimizer can be developed through real-time applications and using various databases for large amounts of subject data especially with several unique facial images such as twins, aging people, and faces that used make up.

Several approaches could be explored to solve the research's issues. In the future, the dataset could be expanded to cover a more diverse individuals, capturing differences in age, gender, and ethnicity, in order to overcome the limitations of a small sample with only 10 face data participants. It would also be possible to alter the camera's fixed position, height, and distance dynamic by implementing multi-angle recordings or adaptive systems to accommodate different user conditions. Although this research focused on one particular CNN architecture, future research could examine how well alternative sophisticated architectures or ensemble approaches perform in order to increase robustness. By incorporating real-

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

world scenarios with different illumination, occlusions, and motion blur, evaluation might be expanded beyond the current measures and implemented to evaluate the system's adaptability and dependability in a range of operating circumstances.

REFERENCES

Alagarsamy, S., Govindaraj, V., Irfan, M., Swami, R., & Kumar, N. M. (2020). Smart Recognition of Real Time Face using Convolution Neural Network (CNN) Technique. *TEST Engineering & Management*, *83*(April), 23406 – 23411.

Alhanaee, K., Alhammadi, M., Almenhali, N., & Shatnawi, M. (2021). Face recognition smart attendance system using deep transfer learning. *Procedia Computer Science*, *192*, 4093–4102. https://doi.org/10.1016/j.procs.2021.09.184

Alzubaidi, L., Zhang, J., Humaidi, A. J., Dujaili, A. Al, Duan, Y., Shamma, O. Al, Santamaría, J., Fadhel, M. A., Amidie, M. Al, & Farhan, L. (2021). Review of deep learning : concepts , CNN architectures , challenges , applications , future directions. In *Journal of Big Data*. Springer International Publishing. https://doi.org/10.1186/s40537-021-00444-8

Amulya, G., Jyothi, A., Dasari, S., Sandhya, E., & V, R. K. K. (2024). Performance Analysis of ML and DL Models : Impact of Linear and Non-Linear Optimizers on Model Efficiency Performance Analysis of ML and DL Models : Impact of Linear and Non- Linear Optimizers on Model Efficiency. *Advances in Nonlinear Variational Inequalities*, *28*(1), 234–250. https://doi.org/10.52783/anvi.v28.2300

Anu, T. A., Rosnelly, R., Irawan, D., Bulolo, P., & Hasibuan, U. (2023). Utilization of Digital Image and Convolution Neural Network Algorithm in Customer Satisfaction Survey with Facial Expressions. *Journal of Computer Science, Information Technology and Telecommunication Engineering*, *4*(2), 420–427. https://doi.org/10.30596/jcositte.v4i2.15915

Backar, S. P., Purnawansyah, P., Darwis, H., & Astuti, W. (2023). Hybrid Fourier Descriptor Naïve Bayes dan CNN pada Klasifikasi Daun Herbal. *Jurnal Informatika: Jurnal Pengembangan IT*, *8*(2), 126–133. https://doi.org/10.30591/jpit.v8i2.5186

Bhatt, H., Shah, V., Shah, K., Shah, R., & Shah, M. (2023). State-of-the-art machine learning techniques for melanoma skin cancer detection and classification: a comprehensive review. *Intelligent Medicine*, *3*(3), 180–190. https://doi.org/10.1016/j.imed.2022.08.004

Cheng, W. C., Hsiao, H. C., & Lee, D. W. (2021). Face recognition system with feature normalization. *International Journal of Applied Science and Engineering*, *18*(1), 1–9. https://doi.org/10.6703/IJASE.202103_18(1).004

Dacipta, P. N., & Putra, R. E. (2022). Sistem Klasifikasi Limbah Menggunakan Metode Convolutional Neural Network (CNN) Pada Webservice Berbasis Framework Flask. *Journal of Informatics and Computer Science (JINACS)*, *3*(04), 394–402. https://doi.org/10.26740/jinacs.v3n04.p394-402

Hanafie, A., Husain, N. P., Kumkelo, H., & Putri, R. R. (2023). Aplikasi Ekstraksi Wajah Menggunakan Algoritma Viola Jones. *ILTEK : Jurnal Teknologi*, *18*(02), 87–91. https://doi.org/10.47398/iltek.v18i02.130

Hasyani, R. A., Simbolon, S. M., Mufida, Y., Ester, Y., & Ritonga, B. (2023). Klasifikasi Malaria melalui Penggunaan Convolutional Neural Network pada Citra Sel Darah. *Student Research Journal*, *1*(6). https://doi.org/10.55606/srjyappi.v1i6.853

Ilahiyah S, & Nilogiri A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12058

Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network _ Ilahiyah _ JUSTINDO (Jurnal Sistem dan Teknologi Informasi Indonesia). *JUSTINDO(Jurnal Sistem & Teknologi Informasi Indonesia)*, *3*(2), 49–56.

Iskandar Zulkarnain Maulana Putra, T., Farhan Bukhori, A., Ilmu Pengetahuan Alam, dan, & Gadjah Mada, U. (2022). Model Klasifikasi Berbasis Multiclass Classification dengan Kombinasi Indobert Embedding dan Long Short-Term Memory untuk Tweet Berbahasa Indonesia (Classification Model Based on Multiclass Classification with a Combination of Indobert Embedding and Long . *Jurnal Ilmu Siber Dan Teknologi Digital (JISTED)*, *1*(1), 1–28. https://doi.org/10.35912/jisted.v1i1.1509

Kurniawan, R., Wintoro, P. B., Mulyani, Y., & Komarudin, M. (2023). Implementasi Arsitektur Xception Pada Model Machine Learning Klasifikasi Sampah Anorganik. *Jurnal Informatika Dan Teknik Elektro Terapan*, *11*(2), 233–236. https://doi.org/10.23960/jitet.v11i2.3034

Lesmana, A. M., Fadhillah, R. P., & Rozikin, C. (2022). Identifikasi Penyakit pada Citra Daun Kentang Menggunakan Convolutional Neural Network (CNN). *Jurnal Sains Dan Informatika*, *8*(1), 21–30. https://doi.org/10.34128/jsi.v8i1.377

Nurdin, A., Satria, D., Kartika, Y., Rezha, A., & Najaf, E. (2024). Klasifikasi Penyakit Daun Tomat Dengan Metode Convolutional Neural Network Menggunakan Arsitektur Inception-V3. *Jurnal Ilmiah Informatika (JIF)*, *12*(2), 114–119.

Purwono, Ma'arif, A., Rahmaniar, W., Fathurrahman, H. I. K., Frisky, A. Z. K., & Haq, Q. M. U. (2022). Understanding of Convolutional Neural Network (CNN): A Review. *International Journal of Robotics and Control Systems*, *2*(4), 739–748. https://doi.org/10.31763/ijrcs.v2i4.888

Putra, C. M., Triayudi, A., & Ningsih, S. (2023). Face Mask Recognition Menggunakan Model CNN (Convolutional Neural Network) Berbasis Python dan OpenCV. *Journal of Computer System and Informatics (JoSYC)*, *4*(3), 722–730. https://doi.org/10.47065/josyc.v4i3.3532

Solihat, S., Widodo, S., & Sari, D. P. (2024). Analisis Perbandingan Optimizer pada Pelatihan Model Convolutional Neural Network untuk Kasus Klasifikasi Hewan Primata. *Jurnal Media Informatika Budidarma*, *8*(1), 459–467. https://doi.org/10.30865/mib.v8i1.7274

Taqiyuddin, M., Adi, K., Dwi Nurhayati, O., & Ochi, H. (2023). Comparison of Optimizers for Drone Signal Detection Using Convolutional Neural Networks (CNN). *E3S Web of Conferences*, *448*. https://doi.org/10.1051/e3sconf/202344802025

Yulina, S. (2021). Penerapan Haar Cascade Classifier dalam Mendeteksi Wajah dan Transformasi Citra Grayscale Menggunakan OpenCV. *Jurnal Komputer Terapan*, *7*(1), 100–109. https://doi.org/10.35143/jkt.v7i1.3411