


Retinopathy Classification using Convolutional Neural Network Method with Adam Optimization and Applied Batch Normalization

Isnandar Slamet^{1,*}, Dhestahendra Citra Susilotomo², Etik Zukhronah³, Sri Subanti⁴, Irwan Susanto⁵,
Winita Sulandari⁶, Sugiyanto⁷, Yuliana Susanti⁸

^{1,2,3,4,5,6,7,8}Department of Statistics, Universitas Sebelas Maret, Jl. Ir. Sutami 36A Kentingan Surakarta 57126, Indonesia

(Received: May 28, 2024; Revised: June 21, 2024; Accepted: July 8, 2024; Available online: July 31, 2024)

Abstract

Retinopathy is a common eye disease in Indonesia, ranking fourth after cataracts, glaucoma, and refractive errors. It can be overcome by early diagnosis with optical coherence tomography (OCT), but this manual reading of imaging technique takes much time. Retinal imaging was carried out to reduce the time using an expert system. The expert system in this study was formed using the convolutional neural network (CNN or ConvNet) method. CNN is an algorithm of deep learning that uses convolution operations to process two-dimensional data, such as images and sounds. This research consisted of 4 stages: data collection, preprocessing, model design, and model testing. Based on the research that has been carried out, a CNN model was formed with three configurations consisting of two convolutional layers and one pooling layer. The ReLU activation function, zero padding, and batch normalization were used in all three formats. Then, the flattening process was carried out, and the Softmax activation function was used at the end of the architecture. The model was built using 8 epochs and Adaptive Momentum (Adam) optimization, resulting in training, validation, and test data accuracy values of 97.32%, 92.64%, and 98.96%. The resulting model can identify retinal diseases with high accuracy, which positively contributes to improving the treatment of retinal diseases.

Keywords: Convolutional Neural Network, Adaptive Momentum, Batch Normalization

1. Introduction

Retinopathy is an eye disease that attacks the retina and can cause vision problems. The causes of retinopathy vary depending on the type. According to the WHO report, the kinds of retinopathy that cause blindness with the highest cases are diabetic retinopathy (DR) and age-related macular degeneration (AMD), with percentages of 1% and 5%, respectively, of the total cases of blindness. Retinopathy can be treated with proper early diagnosis using optical coherence tomography (OCT), whose manual reading of images by medical personnel takes much time. Therefore, with sophisticated technological developments and the strong computing capabilities of computers, it is possible to have an expert system. For experts, using expert systems also helps their activities as experienced assistants.

Artificial intelligence (AI) is an area of research that plays a critical role in developing expert systems. Ghorbanzadeh et al. [1] define machine learning (ML) as an AI approach frequently used to mimic human behavior when completing or automating tasks. Deep learning (DL) is a subfield of ML in which the algorithms are inspired by how the human brain works. Some people have heard of artificial neural networks. Because of its exceptional performance, DL has recently been the focus of ML development. This is influenced by more powerful computational factors, large datasets, and techniques for training deeper networks. The DL method currently being developed is the convolutional neural network (CNN). CNN is a multilayer perceptron (MLP) invention commonly used to analyze image data. Research on the implementation of CNN can be found in Trnovszky et al. [2], Nugroho et al. [3], Krizhevsky et al. [4], Ge et al. [5], Nielsen [6], Kurt [7], and Fukushima [8]. MLP is less accurate in some circumstances regarding image categorization because it does not preserve spatial information and treats each pixel as a separate feature [9].

Srinivasan et al. [10] investigated using OCT images in diagnosing and analyzing AMD and DME. This investigation used the support vector machine (SVM) approach to extract a histogram of oriented gradient (HOG) characteristics from OCT images. Furthermore, Trnovszky et al. [4] compared many image-processing algorithms, including CNN,

*Corresponding author: Isnandar Slamet (isnandarlamet@staff.uns.ac.id)

 DOI: <https://doi.org/10.47738/jads.v5i3.309>

This is an open access article under the CC-BY license (<https://creativecommons.org/licenses/by/4.0/>).

© Authors retain all copyrights

principal component analysis (PCA), linear discriminant (LDA), local binary patterns histograms (LBPH), and SVM. The results showed that the CNN method gave the best results with 98% accuracy. Kermayn et al. [11], in their study, found a multi-class comparison between CNV, DME, Drusen, and Normal with an accuracy of 96.6%, a sensitivity of 97.8%, a specificity of 97.4%, and an error of 6.6%.

Krizhevsky et al. [4] conducted research by training a CNN to classify 1.2 million ImageNet LSVRC-2010 images into 1000 different classes, and the test data had the smallest error compared to other methods. Trnovszky et al. [2] conducted research comparing several image processing methods, namely CNN, PCA, LDA, LBPH, and SVM. The research results show that the CNN method provides the best results with an accuracy of 98%. Abdolamanafi et al. [12] also conducted research comparing three classification methods, namely CNN, random forest (RF), and SVM on Kawasaki disease, which attacks children; the result was that CNN was the best result with an accuracy of 92%

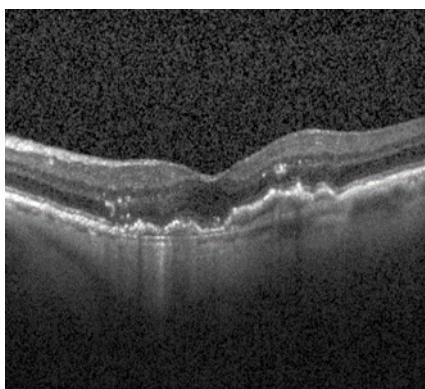
Danukusumo et al. [13] researched classifying GPU-based temple images using the CNN method. The research results show an accuracy of 98.99% on training data and 85.57% on test data, with a training time of 389.14 seconds. In [3], research to identify skin diseases was done using the CNN method on the HAM 10000 dataset. The research results obtained accuracy from the training, validation, and test processes of 83%, 85%, and 85.5%, respectively. This study uses the CNN method to identify retinopathy in OCT image data. Many researchers have researched retina OCT image data. We refer to Hamet and Tremblay [14], Srivastava [15], Mikolajczyk and Schmid [16], and Iofee and Szeged [17] as examples. In Hamet and Tremblay [14], identification was made by classifying retinal diseases into four classes: choroidal neovascularization (CNV), diabetic macular edema (DME), Drusen, and Normal.

2. The Materials and Methods

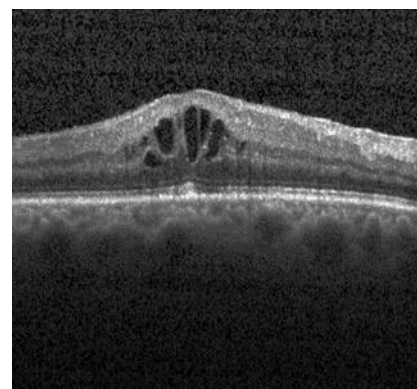
The data used were secondary ones obtained from Kaggle.com, consisting of 4 classes: CNV, DME, Drusen, and Normal. This study's secondary data, retinal OCT, is obtained from Kaggle.com. Retinal OCT is a non-invasive imaging technique that allows for high-resolution cross-sectional imaging of the retina, which is the light-sensitive layer at the back of the eye. It utilizes light waves to capture detailed images of the retina's layers and structures, providing valuable information for diagnosing and managing various eye conditions. The dataset is organized into three folders, i.e., train, test, and val, with subfolders for each picture category. There are 84,495 JPEG X-ray images in four categories, i.e., CNV, DME, drusen, and normal. Images are tagged with (disease)-(randomized patient ID)-(image number by this patient), then arranged into 4 categories: CNV, DME, drusen, and normal.

2.1. Data Preprocessing

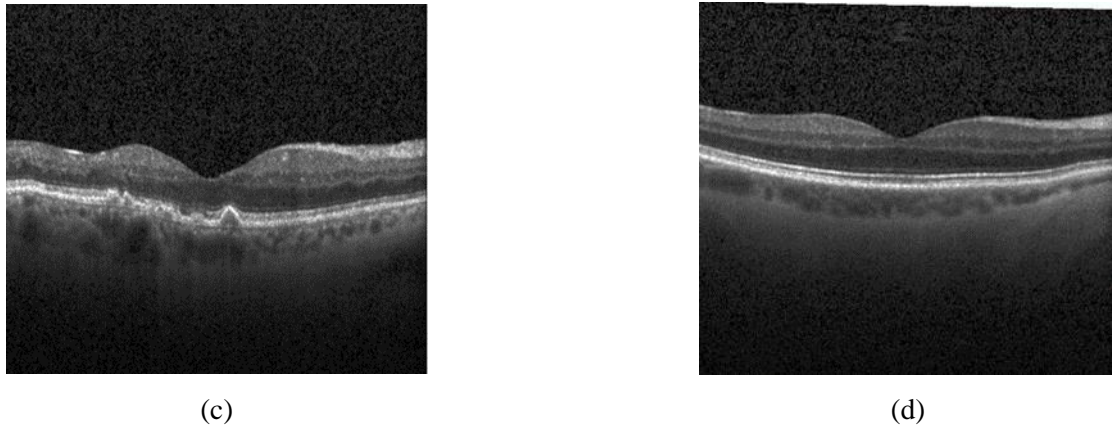
In preprocessing, the data were grouped into four classes: CNV, DME, Drusen, and Normal. After that, the data were divided into training, validation, and test data. In this research, optical coherence tomography image data amounted to 84,495 data. The data was separated into four classes: CNV, DME, drusen, and normal. Figure 1 shows examples of CNV, DME, drusen, and normal images.



(a)



(b)



(c) (d)

Figure 1. (a). CNV (b). DME (c). Drusen (d). Normal

2.2. CNN Model Design

The convolutional neural network is one of the DL algorithms used to process data in two-dimensional forms, such as images and sounds. Based on its name, CNN is a neural network that uses a mathematical operation, namely convolution. CNN is an extension of the MLP. MLP, in some cases, is less accurate in image classification because it assumes that each pixel is an independent feature and does not store spatial information from image data, resulting in poor results [9]. CNN is a layer of 3-dimensional neurons (width, height, depth). Width and height represent layer sizes, while depth refers to the number of layers. In general, the CNN model design was built with several layers. The layers in CNN were divided into convolutional layers, pooling layers, and fully connected layers. Figure 2 presents CNN Architecture.

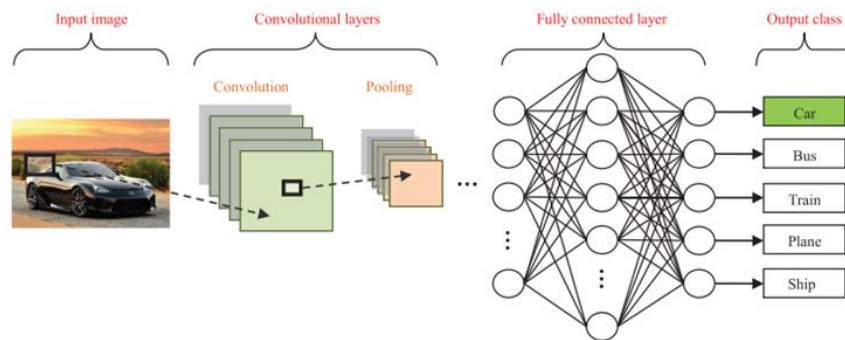


Figure 2. CNN Architecture

The convolutional layer is the layer that receives the input image first. At this stage, the convolution operation was performed. The equation for convolution operation is as follows:

$$s(t)=(x*w)(t) \tag{1}$$

$s(t)$ in (1) notes a convoluted function called a feature map; x denotes the input, and w denotes the kernel. If the input was a two-dimensional image, we could assume t was a pixel and replace it with i and j . Therefore, the operation for convolution to input with two dimensions is presented in the following formula:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i - m, j - n)K(m,n) \tag{2}$$

Equation (2) is a basic calculation in the convolution kernel operation where i and j are the pixels of the image. The analysis is commutative, where I is the input, and K is the kernel.

The pooling layer is a matrix size using the merge operation. Pooling layers will reduce the spatial size and number of parameters in the network to speed up the computing process and control overfitting. A sliding window is applied with

a kernel size of 2x2 and a stride value of one in this process. The method used is max pooling, which means that each window in the feature map matrix resulting from the convolution will take the largest value.

The pooling layer is usually located after the convolution layer and consists of a filter and a particular stride that shifts throughout the feature map area. The layer reduces the spatial size and the number of parameters in the network to speed up the computing process and control overfitting. Figure 3 presents the pooling layer process. Two types of pooling commonly used are max pooling and average pooling [18], [19], [20].

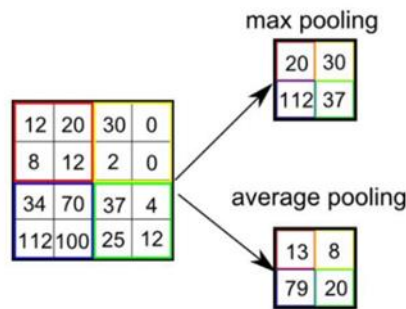


Figure 3. Pooling Layer Process

The fully connected layer process is a process where the feature map is changed to flatten or vector. In this model, 160,000 flattened neurons will be input to the fully connected layer, while the number of neurons in the hidden layer used is 128 neurons. Figure 4 shows the architecture formed in a fully connected network. Generally, these networks should be fully connected, where each pixel is considered a separate neuron. However, a dropout method is applied to avoid overfitting so that several edges connected to each neuron are disabled. The dropout process is carried out randomly according to the random value used. In the fully connected layer, a dropout value of 0.5 is used, meaning that half of all the edges in that layer are disabled.

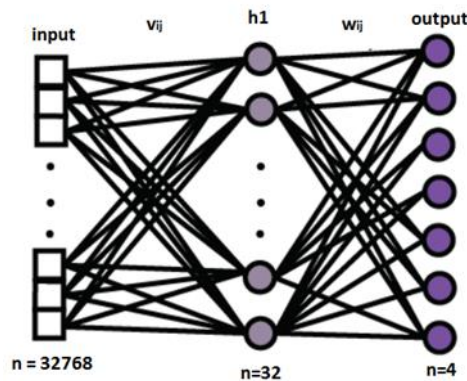


Figure 4. Fully Connected Layer

Generally, these networks should be fully connected, where each pixel is considered a separate neuron. However, a dropout method is applied to avoid overfitting so that several edges connected to each neuron are disabled. The dropout process is carried out randomly according to the random value used. In the fully connected layer, a dropout value of 0.5 is used, meaning that half of all the edges in that layer are disabled. The ReLU activation function is applied to the hidden layer, and Softmax is applied to the output layer in the fully connected layer.

The fully connected layer is where all activation neurons from the previous layer are connected to the neurons of the next layer. Each activation of the prior layer must be converted into one-dimensional data to connect to all neurons in the fully connected layer. This layer is usually used in the MLP method to process data so that it can be classified. The difference between the fully connected layer and the convolutional layer lies in the neurons; the convolutional layer is connected only to some regions of the input, while the fully connected layer is connected as a whole. Since the two layers still operate the dot operation, their function is not that different.

2.3. Comparison of CNN Models

In comparing models, two optimizations are used to find the best result in this study: Stochastic Gradient Descent (SGD), as in Kingma and Adam [18] and Adam. In addition, the model has been implemented using batch normalization and without batch normalization. SGD is a variation of gradient descent optimization that always updates parameters. SGD does not repeat itself when updating parameters, so its performance is faster for large datasets. An important parameter for the SGD algorithm is the learning rate. SGD uses a fixed learning rate η . In practice, it is necessary to carry out SGD gradually to reduce the learning rate over time. The learning rate at iteration k is expressed as η_k [11].

Adam is another adaptive learning rate optimization algorithm [12]. Adam is a combination of RMSProp and momentum with some crucial differences. First, in Adam, momentum is combined directly to approximate the gradient's first-order moment (with exponential weight). Second, Adam incorporates bias corrections into the estimates of first-order and second-order moments to account for early initialization. Batch normalization has been proven effective in reducing the number of epochs required to train neural networks [13]. Batch normalization speeds up network training by normalizing the activations of an input volume before passing it to the next layer. The goal is to reduce covariate shifts. The weakness of batch normalization is that it can slow down the time to train a network (although it requires fewer epochs to get maximum accuracy).

2.4. Testing CNN Model

The confusion matrix in Figure 5 is usually used to calculate accuracy in decision support systems. In this measurement, four terms represent the classification results. The four terms are true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 5. Confusion Matrix

Accuracy is the ratio of correct predictions, both positive and negative, on the overall data, which the following equation can calculate as in (3).

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

3. Results

3.1. Data Preprocessing

The data downloaded from Kaggle.com was then preprocessed by dividing the data into three parts: training, validation, and test data. We do not provide any information on whether data augmentation techniques were used because, in this work, we do not consider data augmentation. We refer to [13] that data augmentation is not necessarily needed for image transformations since the orientation change does not affect this algorithm. We have added the normalization methods in data preprocessing, i.e., Pixel Scaling. Data normalization aims to ensure that all variables in the dataset have the same scale and that no variable dominates the analysis process. This scaling process is carried out to change each pixel value from the range [0.255] to [0.1]. Table 1 presents the type of data.

Table 1. Data Type

Class Type	Data Type		
	Training Data	Validation Data	Test Data
CNV	29764	7441	243
DME	9079	2269	243
Drusen	6893	1723	243
Normal	21052	7441	243

3.2. CNN Model Design

The preprocessing data was generated by rescaling the training, validation, and test data. Rescale was done to change each pixel value from the range [0,255] to [0,1]. The CNN architecture used in this system is shown in Table 2 This architecture used an image size of 128x128x3 as input. This means that the image size used was 128x128 pixels, while the number three indicates the number of channels: red, green, and blue (RGB).

Based on Table 2, the CNN architecture was formed by making layers consisting of one convolution layer and one pooling layer of three layers. The first layer used a filter size of 32, kernel 3x3, and pooling 2x2. The second layer used a filter size of 64, kernel 3x3, and pooling 2x2. The third layer used a filter size 128, kernel 3x3, and pooling 2x2. The ReLU activation function and zero padding were used in these three layers. ReLU was used to change the negative output value to zero. Zero padding was used to produce an output equal to the input size. Then, the flattening process was done by changing the feature map into vector form and dropping out. At the end of the architecture, the Softmax activation function is used to classify it into four classes.

Table 2. Architectural Design of CNN

No	Name	Size	Parameter
1	Input	128*128*3	0
2	Conv2d_1	128*128*32	896
3	MaxPooling2d_1	64*64*32	0
4	Conv2d_2	64*64*64	18.496
5	MaxPooling2d_	32*32*64	0
6	Conv2d_3	32*32*128	73.856
7	MaxPooling2d_3	16*16*128	0
8	Flatten	32,768	0
9	Dense_1	32	1.048.608
10	Dropout	32	0
11	Dense_2	4	132
Total			1.141.988

3.3. Comparison of CNN Models

The CNN model that has been formed is implemented on training data and validation data to train the model to recognize patterns. Two optimizations were used to find the best in this research: Adam and SGD. The first optimization used is Adam with normalization by initializing hyperparameters in the form of learning rate $\eta = 0.001$, exponential decay rate $\rho_1 = 0.9$ and $\rho_2 = 0.999$, small constant $\delta = 10^{-7}$ and parameters in the form of weight and bias.

At this stage, a convolution process occurs for each channel and image. The convolutions at this stage correspond to the filter size used, namely 32 in the first configuration, 64 in the second configuration, and 128 in the third configuration.

In the next stage, padding is carried out so that the output matrix values have the same size as the input matrix. In addition, the model will be implemented using batch normalization and without batch normalization. A comparison of models can be seen in [Table 3](#).

Table 3. Model Comparison

Parameter	Model			
	Adam without normalization	Adam with normalization	SGD without normalization	SGD with normalization
Epochs	6	8	6	8
Accuracy	88.28%	97.32%	77.88%	93.39%
Validation Accuracy	88.98%	92.64%	79.49%	91.16%
Loss	0.3319	0.0854	0.6037	0.2055
Validation Loss	0.3277	0.2383	0.5357	0.2812

[Table 3](#) shows the output of each model that has been tested. In the training data, the highest accuracy is 97.32%, and the lowest loss value is 0.0854, using the normalized Adam model for each layer. In the validation data, the highest accuracy is 92.64%, and the lowest loss value is 0.2383 using the same model. From these results, the Adam model was chosen for normalization. The architectural design was carried out to conduct CNN model training. The accuracy values on the training data and validation data showed how well the model was used on the test data. The training process used an epoch value of 8 and was repeated eight times to train the model.

The choice of eight epochs for training has been made for the case of Adam optimization with batch normalization. [Figure 6](#) shows the model's accuracy using Adam optimization with batch normalization for each layer. The accuracy of the training and validation data experiences an upward trend pattern from the first epoch to the last epoch. The model experiences convergence with Adam optimization at the eighth epoch. The accuracy of the training data when converging is 97.32%, and the accuracy of the validation data when converging is 92.64%. [Figure 7](#) shows each epoch's accuracy and loss values, which converge with the eighth epoch. The model converges at the eighth epoch because the difference between the loss value at the ninth and eighth epoch is less than 0.01.

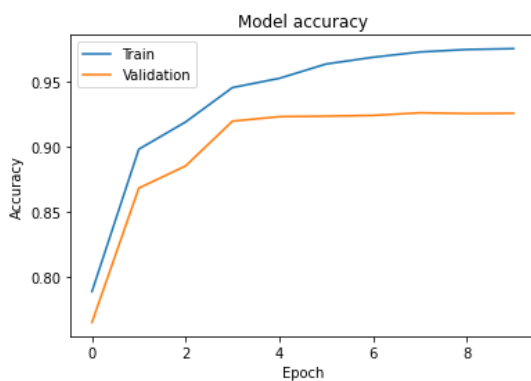


Figure 6. Model's accuracy using Adam optimization with batch normalization

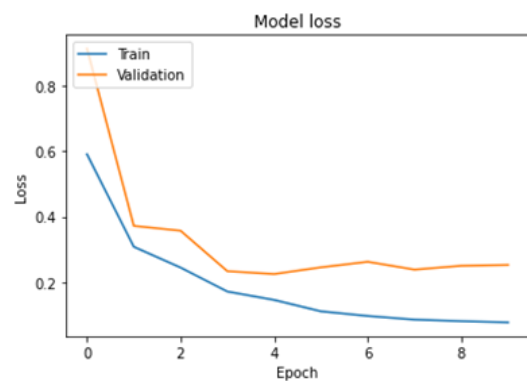


Figure 7. Plot loss and best model loss values

3.4. Testing CNN Model

Model testing was performed on 972 test data sets, with 243 data sets for each class. [Table 4](#) depicts the results of the testing process. [Figure 8](#) explains that the model that has been built can correctly classify CNV with as much as 241 data, DME with as much as 237 data, Drusen with as much as 239 data, and Normal with as much as 241 data. Therefore, the accuracy of the test data obtained is 98.96%.

Table 4. Accuracy and loss for each Adam epoch with batch normalization

Epochs	Loss	Accuracy	Val loss	Val accuracy
1	591.061	789.053	913.953	765.355
2	308.172	898.421	372.094	868.522
3	244.518	919.393	357.323	885.617
4	171.355	945.803	233.443	919.986
5	14.566	952.903	224.821	923.464
6	110.782	963.928	245.245	923.824
7	96.547	969.081	262.277	924.424
8	85.444	973.156	238.348	926.404
9	80.741	975.058	250.011	925.864
10	76.954	975.718	252.637	926.044

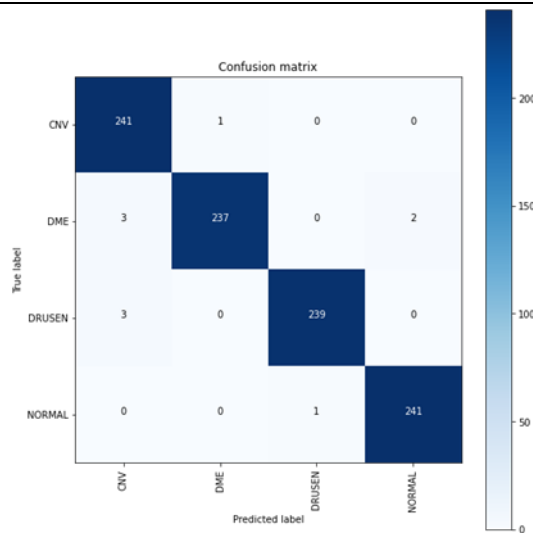


Figure 8. Confusion Matrix

Table 5 presents precision, recall, and F1-score information for all classes. The table presents the performance metrics for a classification task involving four classes: CNV, DME, DRUSEN, and Normal. Each class is evaluated using precision, recall, and F1-score percentages, key indicators of the model's effectiveness. The CNV class has high precision (97.6%) and excellent recall (99.6%), leading to an impressive F1-score of 98.6%, indicating the model's reliability in correctly identifying CNV instances. The DME class shows the highest precision at 99.6%, although its recall is slightly lower at 97.9%, resulting in an F1-score of 98.7%. DRUSEN class achieves balanced precision and recall at 98.8%, yielding an F1-score of 98.8%, which suggests consistent performance in identifying DRUSEN cases. The Normal class exhibits perfect precision at 100% and near-perfect recall at 99.6%, with an outstanding F1 score of 99.8%, highlighting the model's accuracy in distinguishing normal instances. Overall, the average performance across all classes is excellent, with 99% precision, recall, and F1-score, indicating a well-performing model across different categories.

Table 5. Performance of Class

Class	Precision (%)	Recall (%)	F1-Score (%)
CNV	97.6%	99.6%	98.6%
DME	99.6%	97.9%	98.7%
DRUSEN	98.8%	98.8%	98.8%

Normal	100%	99.6%	99.8%
Average	99%	99%	99%

4. Discussion

Based on the discussion, it can be concluded that the best CNN model was formed with three layers consisting of one convolutional layer and one pooling layer. ReLU activation functions, zero padding, and batch normalization were used in all three layers. The flattening and dropout processes were then performed, and the SoftMax activation function was used at the end of the architecture. The next step is to change the color pixels to numeric. Converting images to numeric is achieved by extracting each channel. After each channel is extracted, a rescaling process changes each pixel value from the range [0.255] to [0.1]. This study illustrates the ability of the CNN model to classify retinal diseases. Two optimizations are used, namely Adam and SGD. Both optimizations are implemented using batch normalization and without batch normalization.

The first model uses Adam optimization without batch normalization for each layer. The accuracy of the training and validation data experiences an upward trend pattern from the first epoch to the last epoch. The model experiences convergence with Adam optimization at the sixth epoch. The accuracy of the training data when converging is 88.28%, and the accuracy of the validation data is 88.98%. The training and validation data loss values experienced a downward trend pattern from the first to the last epochs. When converging, the training data loss value is 0.3319, and the validation data loss value is 0.3277.

In the second optimization, namely Adam, without batch normalization for each layer, the accuracy of the training and validation data experienced an upward trend pattern from the first epochs to the last epochs. The model experiences convergence with Adam optimization at the eighth epoch. The accuracy of the training data when converging is 97.32%, and the accuracy of the validation data when converging is 92.64%. The loss value from the training data experienced a downward trend pattern from the first epochs to the last epochs, while the validation data experienced fluctuations. When converging, the training data loss value is 0.0854, and the validation data loss value is 0.2383. The accuracy and loss values of each epoch converge at the eighth epoch. The model converges at the eighth epoch because the difference between the loss value at the ninth and eighth epoch is less than 0.01.

In optimization using SGD, initialize hyperparameters in the form of learning rate $\eta = 0.01$ and parameters in the form of weights and bias. The model stops with this optimization when its maximum epoch is ten epochs. The model converges when the loss value does not increase or decrease by less than 0.01. In this model, batch normalization is not applied to each layer, so the input volume is not normalized before proceeding to the next layer. The accuracy of the training and validation data experienced an upward trend pattern from the first epochs to the last epochs. The model experiences convergence with SGD optimization at the sixth epoch. The accuracy of the training data when converging is 77.88%, and the accuracy of the validation data when converging is 79.49%. The training and validation data loss values experienced a downward trend pattern from the first to the last epochs. When converging, the training data loss value is 0.6037, and the validation data loss value is 0.5357. The model converges at the sixth epoch because the difference between the loss value at the seventh and sixth epochs is less than 0.01.

For SGD Optimization with batch normalization for each layer, the input volume is normalized before proceeding to the next layer. The accuracy of the training data experienced an upward trend pattern from the first epochs to the last epochs, while the validation data experienced fluctuations. The model experiences convergence with SGD optimization at the eighth epoch. The accuracy of the training data when converging is 93.39%, and the accuracy of the validation data when converging is 91.16%. The loss value from the training data experiences a downward trend pattern from the first epochs to the last epochs, while the validation data experiences fluctuations. When converging, the training data loss value is 0.2055, and the validation data loss value is 0.2812. The accuracy and loss values of each epoch converge at the eighth epoch.

On the training data, the highest accuracy was 97.15%, and the lowest loss value was 0.0899 using the Adam model, which was normalized for each layer. In the validation data, the highest accuracy was 92.62%, and the lowest loss value was 60.2581 using the same model. From these results, the Adam model with normalization was chosen. The

model built correctly classified 241 data of CNV, 237 data of DME, 239 data of Drusen, and 241 data of Normal. The test data accuracy rate is 98.96%. This accuracy value is very good and very close to the accuracy value of the training data, namely 97.32%

5. Conclusion

Based on the research results and discussions that have been carried out, it can be concluded that the best CNN model is formed with three arrangements consisting of one convolutional layer and one pooling layer. The ReLU activation function, zero padding, and batch normalization are used in these three configurations. Furthermore, the flatten dropout process is carried out and the Softmax activation function is used at the end of the architecture. The model was built using eight epochs with Adam optimization with batch normalization, which produced training, validation, and test data accuracy values of 97.32%, 92.64%, and 98.96%.

The proposed method can classify the four studied retinal diseases with high accuracy. One advantage of the method is that the Adam optimizer computes adaptive learning rates for each parameter, allowing for faster convergence and better handling of sparse gradients, which is common in medical image datasets. Using batch normalization across all network layers is recommended, as it makes a significant difference. Applying batch normalization to a network architecture can help prevent overfitting and make it possible to obtain higher classification accuracy in fewer epochs compared to the same network architecture without batch normalization.

6. Declarations

6.1. Author Contributions

Conceptualization: I.S., D.C.S., E.Z., S.S., I.S., W.S., S., and Y.S.; Methodology: E.Z., I.S., W.S., and S.; Software: I.S., D.C.S., and E.Z.; Validation: I.S., S.S., and Y.S.; Formal Analysis: I.S., D.C.S., and E.Z.; Investigation: I.S., D.C.S., and E.Z.; Resources: E.Z., S.S., I.S., and W.S.; Data Curation: E.Z., S.S., and I.S.; Writing Original Draft Preparation: I.S., D.C.S., and E.Z.; Writing Review and Editing: S.S., I.S., W.S., and S.; Visualization: I.S., D.C.S., and E.Z.; All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

We thank Universitas Sebelas Maret, Surakarta, Indonesia, for providing the funds under Research Group Grant No. 194.2/UN27.22 /PT. 01.03/2024.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] O. Ghorbanzadeh, T. Blaschke, K. Gholamnia, S. R. Meena, D. Tiede, and J. Aryal, "Evaluation of Different Machine Learning Methods and Deep-Learning Convolutional Neural Networks for Landslide Detection," *Remote Sensing*, vol. 11, no. 2, pp. 196-210, Jan. 2019, doi: 10.3390/rs11020196.
- [2] T. Trnovszky, P. Kamencay, R. Orjesek, M. Benco, and P. Sykora, "Animal Recognition System Based on Convolutional Neural Network," *Advances in Electrical and Electronic Engineering*, vol. 15, no. 3, pp. 1-12, Oct. 2017, doi:

10.15598/aeee.v15i3.2202.

- [3] A. A. Nugroho, I. Slamet, and Sugiyanto, "Skins cancer identification system of HAMI0000 skin cancer dataset using convolutional neural network," in *International Conference on Science and Applied Science (ICSAS)*, vol. 2019., no. Dec., pp. 1-6, 2019, doi: 10.1063/1.5141652
- [4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2012, doi: 10.1145/3065386.
- [5] L. Ge, R. Hang, Y. Liu, and Q. Liu, "Comparing the Performance of Neural Network and Deep Convolutional Neural Network in Estimating Soil Moisture from Satellite Observations," *Remote Sensing*, vol. 10, no. 9, pp. 1327-1336, Aug. 2018, doi:10.3390/rs10091327.
- [6] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.
- [7] I. Kurt, M. Ture, and A. T. Kurum, "Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease," *Expert Systems with Applications*, vol. 34, no. 1, pp. 366–374, Jan. 2008, doi: 10.1016/j.eswa.2006.09.004.
- [8] K. Fukushima, "Neocognitron: Deep Convolutional Neural Network," *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics*, vol. 27, no. 4, pp. 115–125, 2015, doi: 10.3156/jsoft.27.4_115.
- [9] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1-12, Jul. 2019, doi: 10.1186/s40537-019-0197-0.
- [10] P. P. Srinivasan, "Fully automated detection of diabetic macular edema and dry age-related macular degeneration from optical coherence tomography images," *Biomedical Optics Express*, vol. 5, no. 10, pp. 3568–3568, Oct. 2014, doi: 10.1364/boe.5.003568.
- [11] D. Kermany, K. Zhang, & M. Goldbaum, Jan 2018, "Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification," Mendeley Data, doi: 10.17632/rschjbr9sj.2
- [12] A. Abdolamanafi, L. Duong, N. Dahdah, and F. Cheriet, "Deep feature learning for automatic tissue classification of coronary artery using optical coherence tomography," *Biomedical Optics Express*, vol. 8, no. 2, pp. 1-9, 2017.
- [13] K. P. Danukusumo, Pranowo, and M. Maslim. "Indonesia ancient temple classification using convolutional neural network," in *2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC)*, Yogyakarta, vol. 1, no. 1, pp.50-54, 2017, doi: 10.1109/ICCEREC.2017.8226709.
- [14] P. Hamet and J. Tremblay, "Artificial intelligence in medicine," *Metabolism: clinical and experimental*, vol. 69S, no. Jan., pp. S36–S40, Jan. 2017, doi:10.1016/j.metabol.2017.01.011.
- [15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, Jan. 2014.
- [16] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005, doi: 10.1109/tpami.2005.188.
- [17] S. Ioffe, and C. Szeged, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, vol. 2015, no. Jun., pp. 448-456, Jun. 2015, doi: 10.48550/arXiv.1502.03167
- [18] D. Kingma, and J. Adam: "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, San Diego, vol. 2015, no. May, pp. 1-13, May. 2015, doi: 10.48550/arXiv.1412.6980
- [19] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional Networks with Dense Connectivity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 1, pp. 1–1, 2019, doi: 10.1109/tpami.2019.2918284.
- [20] T. Schaul, S. Zhang, and Y. LeCun, "No More Pesky Learning Rate," in *International Conference on Machine Learning*, vol. 2013, no. May, pp. 343-351, May. 2013, doi: 10.48550/arXiv.1206.1106