ICIC Express Letters
Volume **14**, Number **7**, July **2020**

ICIC International ©2020 ISSN 1881-803X
pp. **635–641**

RDCNET: CONVOLUTIONAL NEURAL NETWORKS FOR CLASSIFICATION OF RETINOPATHY DISEASE IN UNBALANCED DATA CASES

Bambang Krismono Triwijoyo[1,2], Boy Subirosa Sabarguna[3]
Widodo Budiharto[1] and Edi Abdurachman[1]

[1]Doctor in Computer Science, BINUS Graduate Program
Bina Nusantara University
Jl. Kebon Jeruk Raya No. 27, Kebon Jeruk, Jakarta 11530, Indonesia
bambang.triwijoyo@binus.ac.id; { wbudiharto; edia }@binus.edu

[2]Department of Computer Science
Faculty of Engineering and Health
University of Bumigora
Jl. Ismail Marzuki, Mataram, Nusa Tenggara Barat 83231, Indonesia
bkrismono@universitasbumigora.ac.id

[3]Department of Community Medicine
Faculty of Medicine
University of Indonesia
Jl. Pegangsaan Timur, Jakarta 10310, Indonesia
sabarguna08@ui.ac.id

Received November 2019; accepted February 2020

Abstract. *Retinopathy disease is a type of retinal disorder, which often occurs, including hypertensive retinopathy and diabetic hypertension. Detection of retinopathy can be by analyzing the retinal image, using a deep learning approach, but the problem that is often faced is unbalanced data. In this study, a convolutional neural network architecture is proposed for the classification of retinopathy using the MESSIDOR database that has been labeled, by duplicating and augmentation of sample images in classes with low numbers of samples using a data generator to overcome the problem of unbalanced data. The experimental results show that the validation and testing accuracy performance on the model with two output classes are 100%, and 87.50%, while on the model with four output classes are 99.38%, and 76.47%.*

Keywords: Deep learning, Convolutional neural network, Retinopathy diseases, Image classification, Unbalanced data

1. **Introduction.** Retinal images are a critical factor for ophthalmologists in the diagnosis of several eye diseases. Retinopathy is one type of disease in the retina of the eye, with retinal microvascular signs, which occurs in response to the presence of high blood pressure or diabetes in the patient [1]. The physical symptoms of retinopathy are narrowing of retinal vessels, while other major signs are retinal hemorrhage and cotton wool spots. Traditionally, ophthalmologists use fundus images or retinal images of the eye, to evaluate the presence of retinopathy and to define the evolutionary phase, but traditional methods have limitations, in the case of early symptoms of retinopathy it will be difficult to identify manually, so often ignored [2].

Research on the identification of retinopathy through retinal image has been done before, such as diagnosis of hypertension retinopathy using multiscale filtering and morphological methods based on the Ratio of Arterial and Venous (AVR) vessels have been performed by [3] and using Radon Transform [4]. While [5] performed a diagnosis of

DOI: 10.24507/icicel.14.07.635

hypertension retinopathy based on arterial and venous features of retinal images using four classification methods: Artificial Neural Networks (ANN), Support Vector Machine (SVM), Naïve Bayes and Decision Tree. These studies still use preprocessing algorithms and feature extraction segmentation, before the classification process.

Research on the classification of diabetic retinopathy has also been proposed by [6] using the Convolutional Neural Network (CNN). It uses 12 convolutional layers, thus involving many parameters in the model which results in greater computational complexity of the model training process. [7] conducted diabetic retinopathy classification using SVM Soft Margin. Classification using the random forest technique based on the area and perimeter of the blood vessels and hemorrhages is proposed by [8]. All of these studies also still use preprocessing algorithms and image feature extraction before the classification process. In this study we applied a deep learning method, in which the process of feature extraction and classification of retinopathy are directly carried out on CNN, which has been widely implemented for image classification, including by [9] to detect plant nutrient deficiency based on plant images, and research by [10] for the classification of shape images.

This paper is organized as follows. After the Introduction section, Section two presents our method of pre-processing the retinal image, CNN architecture for classification of retinopathy diseases and solutions to deal with class imbalances. Section three shows the experiments and results. Finally, the fourth section will conclude the study.

2. Methodology.

2.1. Convolutional Neural Network (CNN).

CNN consists of various layers and several neurons on each layer. Both of these are difficult to determine using definite rules and apply differently to different data [11]. CNN operates in the sequence layer by layer, as illustrated in Figure 1 and Table 1 shows the detailed configuration of the deep learning model for retinopathy classification which is the adoption and development of research by [12].

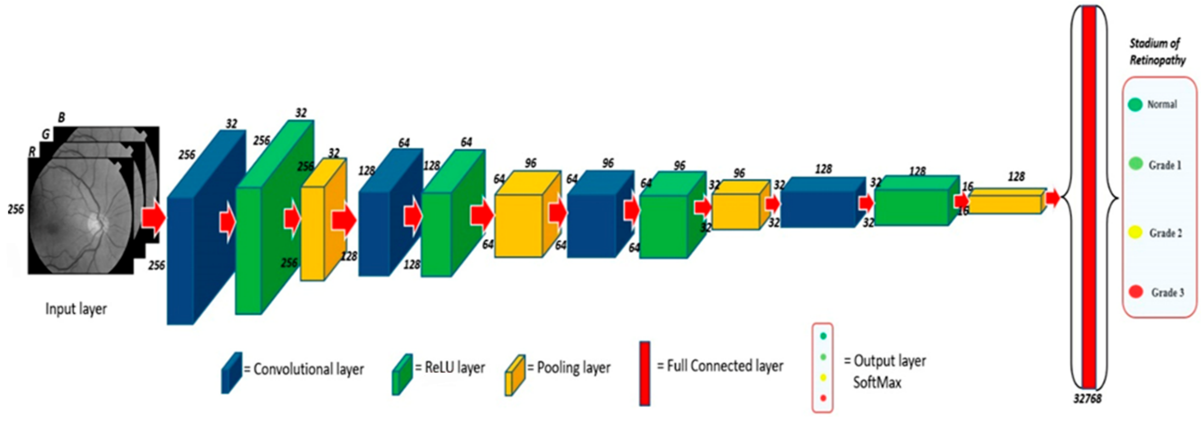


Figure 1. (color online) Model architecture

2.1.1. *Input layer.* Input layer x^l in the form of a 3rd order tensor, where $x^l \in \mathbb{R}^{H_l \times W_l \times D_d}$ is a representation of the colored image of the size of H row, column W, and D color channels. In this case $H = 256$, $W = 256$, and there are three channels of red canal (R), green channel (G) and blue channel (B), so the number of image elements is $256 \times 256 \times 3$ and each element is designated by index (i, j, d), where $0 \leq i < H$, $0 \leq j < W$ and $0 \leq d < 3$.

Table 1. Model configuration

No	Layer	Number of neurons	Padding	Number of kernels	Filter kernel size	Stride
1	Input	$256 \times 256 \times 3$	–	–	–	–
2	Convolutional	256×256	2	32	3×3	1
3	ReLU	256×256	–	32		
4	MaxPool	128×128	–	32	2×2	2
5	Convolutional	128×128	2	64	3×3	1
6	ReLu	128×128	–	64		
7	MaxPool	64×64	–	64	2×2	2
8	Convolutional	64×64	2	96	3×3	1
9	ReLu	64×64	–	96		
10	MaxPool	32×32	–	96	2×2	2
11	Convolutional	32×32	2	128	3×3	1
12	ReLU	32×32	–	128		
13	MaxPool	16×16	–	128	2×2	2
14	Full-Connected	$16 \times 16 \times 128 = 32,768$				
15	Output Softmax	2 or 4				

2.1.2. *Convolutional layer.* The convolutional layer w^l uses multiple convolutional kernels. It assumed the kernel D and each kernel of $H \times W$ used, all kernels denoted as $\boldsymbol{f}$, where $\boldsymbol{f}$ is a 4th order tensor with $\mathbb{R}^{H \times W \times D^l \times D}$ and the index variable $0 \leq i < H$, $0 \leq j < W$, $0 \leq d^l < D^l$ and $0 \leq d < D$ are used to point to one of the kernel elements.

Stride (s) is the concept of the convolution process, where if the value of $s = 1$, then the convolution process is carried out using a kernel matrix size $H \times W$ that shifts to each pixel location of the input image, whereas if the value of $s > 1$, then the distance is shifted by s pixel. The convolution process is expressed through the following equation:

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=0}^{H} \sum_{j=0}^{W} \sum_{d^l=0}^{d^l} f_{i,j,d^l,d} \times x^l_{i^{l+1}+i,j^{l+1}+j,d^l} + b_d \tag{1}$$

for all $0 \leq d \leq D = D^{l+1}$, as well as for any spatial location $\left(i^{l+1}, j^{l+1}\right)$ for $0 \leq i^{l+1} < H^l - H + 1 = H^{l+1}$, $0 \leq j^{l+1} < W^l - W + 1 = W^{l+1}$ and $x^l_{i^{l+1}+i,j^{l+1}+j,d^l}$ refers to elements of x^l at locations with indices $\left(i^{l+1} + i, j^{l+1} + j, d^l\right)$. The bias constant (b_d) is added to Equation (1) with a value of 1.

2.1.3. *ReLU layer.* The ReLU layer does not change the input size, where x^l and y are the same size. The Rectified Linear Unit (ReLU) layer can be considered as the transfer function of each of the input elements as:

$$y_{i,j,d} = \max\left\{0, x^l_{i,j,d}\right\} \tag{2}$$

where $0 \leq i^{l+1} < H^l = H^{l+1}$, $0 \leq j < W^l = W^{l+1}$ and $0 \leq d < D^l = D^{l+1}$, within the ReLU layer, there is no learning parameter as found in the pooling layer.

2.1.4. *Pooling layer.* The pooling operator maps each subpart into a single value. This study used max pooling, where the maximum pooling operator maps the sub-section to the largest value of the element in the sub-section. The following is the mathematical equations of max pooling:

$$\text{max: } y_{i^{l+1},j^{l+1},d} = \max_{0 \leq i < H, 0 \leq j < W} x^l_{i^{l+1} \times H + i, j^{l+1} \times W + j, d^l} \tag{3}$$

where $0 \leq i^{l+1} < H^l$, $0 \leq j^{l+1} < W^{l+1}$ and $0 \leq d < D^{l+1} = D^l$.

2.1.5. *Fully connected layer.* Fully connected layer is a layer where there exists a calculation relationship of each element in the input layer x^l to each element of the output layer x^{l+1} or y. In the CNN model, the fully connected layer is located between the convolutional layer and the output layer.

2.1.6. *Output layer.* The output layer present in the last layer of CNN to the normalized exponential function or softmax is a generalization of the logical function of a k-dimensioned z vector into a k-dimensioned $\sigma(z)$ vector with a real number value between $[0, 1]$. The softmax function is written in the following equation:

$$\sigma : \mathbb{R}^K \to [0, 1]^K \tag{4}$$

$$\sigma(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \ldots, K \tag{5}$$

where σ is softmax notation symbol, z is a vector of the inputs to the output layer, K is dimensions of vector z, and j is the index of the output unit. Table 1 shows the specifications of the model configuration.

2.1.7. *Computational complexity.* Referring to [13-15], the total computational complexity of the model is shown in the following equation:

$$o\left(\left(\sum_{i=1}^{d} n_{i-1} s_i^2 n_i m_i^2\right) + \left(\sum_{j=1}^{l} k_j \log k_j\right)\right) \tag{6}$$

where i is the index of the convolutional layer, and d is the depth or number of convolutional layers. n_i is the number or width of the filter in the ith layer. n_{i-1} is the number of input channels of the ith layer. s_i is the spatial size or length of the filter. m_i is the spatial size of the output feature map. l is the number of fully-connected layers and k_j is the number of nodes in the jth fully-connected layer, including the output layer. The computational complexity of the model becomes a reference in the design of the classification model, although the actual running time is very sensitive to the implementation and environment of the hardware system used.

2.2. **Dataset.** In this study, we used input data from MESSIDOR (Methods to evaluate segmentation and indexing techniques in the field of retinal ophthalmology) [16]. MESSIDOR database consists of 1200 eye fundus color digital images saved as uncompressed TIFF format, 588 images with dimensions of 1440 × 960 pixels, 400 images with dimensions of 2240 × 1488 pixels and 212 images with dimensions of 2304 × 1536. Every image has been labeled by the medical experts into 4 class labels [17].

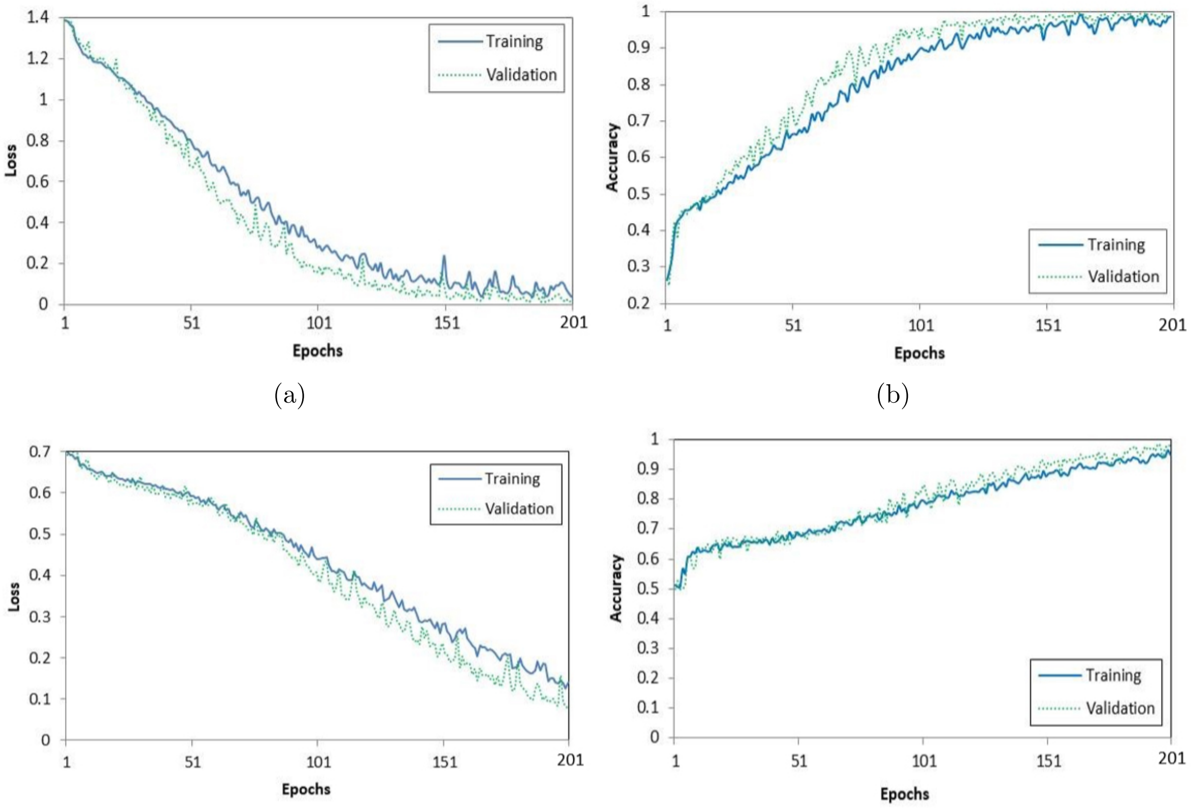
Table 2 shows the details of class labeling and the number of images for each class according to the annotations specified in the MESSIDOR database. The number of images for each class is not balanced, and then in some the class is reduced and added by duplicating and augmenting the image in the same class. Resizing input images is needed to reduce the complexity of input data. In this study, all input images were resized to 256 × 256 pixels using Bicubic Interpolation.

3. Experiments and Results.

3.1. **Training network.** The number of training data is 1200 images, 720 images for training, and 480 images for validations. The dimension of the image input on this model is 256 × 256 pixels. The batch size is 16 and the learning rate value is 0.0001. Then the loss function uses Adam optimization. The image augmentation process in this study was used by changing the scale of image input pixel values from the range $[0, \ldots, 255]$ to $[0, \ldots, 1]$. Then the image is shifted and scaled with a range of shear and zoom values of 0.2, then rotated counterclockwise and enlarges the image to produce new image data that is different from the original image input. The batch size is 16, where 16 training

Table 2. Number of images for each class

	MESSIDOR database		Data training set for 4 classes			Data training set for 2 classes			
Category	Class label	Number of images	Used	Number of duplicated images	Number of images	Class label	Used	Number of duplicated images	Number of images
Normal	0	546	300	0	300	0	546	56	600
Retinopathy Grade 1	1	153	153	147	300		140	0	
Retinopathy Grade 2	2	247	247	53	300	1	227	0	600
Retinopathy Grade 3	3	254	254	46	300		233	0	



(a)

(b)

(c)

(d)

Figure 2. (a) Loss, (b) accuracy of the model with 2 output classes and (c) loss, (d) accuracy of the model with 4 output classes

data are taken randomly from all sample datasets for each epoch until all epochs reach the sample limit.

Training is executed on a computer with specifications processor Intel Core i7-7500U processor specifications, 12 GB RAM, GPU: NVIDIA GeForce GTX 960, Windows 10 operating system, Python 3.6 Programming Language with an editor Jupyter notebook. Figure 2 shows the trend loss and accuracy of the training process and the validation of the two models is almost the same, where up to 200 epochs, in the model with 2 output classes, the loss in the training process is 6.89%, and loss in the validation process is 2.38%, the accuracy of the training process is 97.50%, and the accuracy of the validation process is 100%. While the model with four output classes, the loss in the training process is 4.13% and the loss in the validation process is 2.18%, the accuracy in the training process is 98.67% and the accuracy of the validation process is 99.38%.

3.2. **Testing model.** We use 30 images as independent sample test data. Model performance measured using a performance matrix using three performance measure parameters, namely Specificity, Accuracy, and Precision [18], each of which is defined as follows:

$$\text{Specificity} = \text{TP}/(\text{TP} + \text{FN}) \tag{7}$$

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \tag{8}$$

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) \tag{9}$$

where True Positive (TP) is image class x is classified as image class x, True Negative (TN) is image non-class x, classified as image non-class x, False Positive (FP) is image non-class x, classified as image class x, False Negative (FN) is image class x, classified as image non-class x.

The Specificity, Accuracy, and Precision values of the model testing result with two output classes are 93.33%, 87.50%, and 93.33%, while the models with four output classes have Specificity, Accuracy, and Precision values being 86.67%, 76.47%, and 86.67% respectively. Table 3 shows a comparison of the performance of retinopathy classifications between the proposed methods and those of other previous researchers. Our method has the highest validation accuracy compared to the previous related work, which is 100% on model with two output classes and 99.38% on model with four output classes. However, testing accuracy only reaches 87.50% on model with two output classes and 76.47% on model with four output classes.

Table 3. Performance comparison of retinopathy classification

Author	Method	Database	Accuracy (%)
Manikis et al. [3]	Multiscale Filtering	DRIVE	93.71
		STARE	93.18
Noronha et al. [4]	Radon Transform	STARE	92.00
Abbasi and Akram [5]	ANN	Local Database	76.00
	SVM		75.00
	Naïve Bayes		68.00
	Decision Tree		81.00
Pratt et al. [6]	CNN	Kaggle	70.00
Tjandrasa et al. [7]	SVM	MESSIDOR	90.54
Jain and Ganotra [8]	Random Forest	STARE	90.00
Proposed Method	CNN	MESSIDOR	87.50

4. Conclusion.

In this paper, we propose CNN architecture for the classification of retinopathy, in the case of unbalanced data, using a database of retinal images labeled from MESSIDOR. Unbalanced data is overcome by oversampling techniques through duplication and augmentation of retinal images in the minority class, as well as undersampling techniques, by selecting some data in the majority class. The experimental results show that a model that uses two output classes produces better validation and testing accuracy than a model with four output classes 100% and 87.50% respectively. Our future work is tuning the model by involving the drop-out process and using different machine learning to get better model performance.

REFERENCES

[1] T. Y. Wong and P. Mitchell, Hypertensive retinopathy, *New England Journal of Medicine*, vol.351, no.22, pp.2310-2317, 2004.

[2] S. Khitran, M. U. Akram, A. Usman and U. Yasin, Automated system for the detection of hypertensive retinopathy, *The 4th International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp.1-6, 2014.

[3] G. C. Manikis, V. Sakkalis, X. Zabulis, P. Karamaounas, A. Triantafyllou, S. Douma, C. Zamboulis and K. Marias, An image analysis framework for the early assessment of hypertensive retinopathy signs, *Proc. of the 3rd International Conference on E-Health and Bioengineering (EHB)*, pp.413-418, 2011.

[4] K. Noronha, K. T. Navya and K. P. Nayak, Support system for the automated detection of hypertensive retinopathy using fundus images, *International Conference on Electronic Design and Signal Processing (ICEDSP)*, pp.7-11, 2015.

[5] U. G. Abbasi and M. U. Akram, Classification of blood vessels as arteries and veins for diagnosis of hypertensive retinopathy, *The 10th International Computer Engineering Conference (ICENCO)*, pp.5-9, 2014.

[6] H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding and Y. Zheng, Convolutional neural networks for diabetic retinopathy, *Procedia Computer Science*, vol.90, pp.200-205, 2016.

[7] H. Tjandrasa, R. E. Putra, A. Y. Wijaya and I. Arieshanti, Classification of non-proliferative diabetic retinopathy based on hard exudates using soft margin SVM, *IEEE International Conference on Control System, Computing and Engineering*, pp.376-380, 2013.

[8] S. Jain and D. Ganotra, Detection and classification of diabetic retinopathy in retinal images using ANN, *International Journal of Scientific Research in Science, Engineering and Technology*, vol.2, no.3, pp.319-326, 2016.

[9] S. Z. Maw, T. T. Zin, M. Yokota and E. P. Min, Classification of shape images using K-means clustering and deep learning, *ICIC Express Letters*, vol.12, no.10, pp.1017-1023, 2018.

[10] L. A. Wulandhari, A. A. S. Gunawan, A. Qurania, P. Harsani, Triastinurmiatiningsih, F. Tarawan and R. F. Hermawan, Plant nutrient deficiency detection using deep convolutional neural network, *ICIC Express Letters*, vol.13, no.10, pp.971-977, 2019.

[11] D. Stathakis, How many hidden layers and nodes?, *International Journal of Remote Sensing*, vol.30, no.8, pp.2133-2147, 2009.

[12] B. K. Triwijoyo, Y. Heryadi, A. S. Ahmad, B. S. Sabarguna, W. Budiharto and E. Abdurachman, Retina disease classification based on colour fundus images using convolutional neural networks, *International Conference on Innovative and Creative Information Technology (ICITech)*, pp.1-4, 2017.

[13] K. He and J. Sun, Convolutional neural networks at constrained time cost, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.5353-5360, 2015.

[14] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary and S. F. Chang, An exploration of parameter redundancy in deep networks with circulant projections, *Proc. of the IEEE International Conference on Computer Vision*, pp.2857-2865, 2015.

[15] M. Sun, Z. Song, X. Jiang, J. Pan and Y. Pang, Learning pooling for convolutional neural network, *Neurocomputing*, vol.224, pp.96-104, 2017.

[16] *MESSIDOR: Methods for Evaluating Segmentation and Indexing Techniques Dedicated to Retinal Ophthalmology*, http://messidor.crihan.fr/index-en.php, 2004.

[17] E. Decencière, X. Zhang, G. Cazuguel, B. Laÿ, B. Cochener, C. Trone and B. Charton, Feedback on a publicly distributed image database: The Messidor database, *Image Analysis & Stereology*, vol.33, no.3, pp.231-234, 2014.

[18] L. Zhang, J. Li, H. Han, B. Liu, J. Yang and Q. Wang, Automatic cataract detection and grading using deep convolutional neural network, *IEEE 14th International Conference Networking, Sensing, and Control (ICNSC)*, pp.60-65, 2017.