Jurnal ELTIKOM:
Jurnal Teknik Elektro, Teknologi Informasi dan Komputer

Vol. 8, No. 1, June 2024, page. 66-74
ISSN 2598-3245 (Print), ISSN 2598-3288 (Online)
DOI: http://doi.org/10.31961/eltikom.v8i1.1112
Available online at http://eltikom.poliban.ac.id

DIABETIC RETINOPATHY SEVERITY LEVEL DETECTION USING CONVOLUTION NEURAL NETWORK

Achmad Dinofaldi Firmansyah[1*], Saliyah Binti Kahar[1], Zilvanhisna Emka Fitri[2]
[1)] Department of Information Sciences and Computing, Management and Science University, Shah Alam, Malaysia
[2)] Department of Information Technology, Politeknik Negeri Jember, Jember, Indonesia
e-mail: aldi.novaldi.frmnsyah@gmail.com, saliyah_kahar@msu.edu.my, zilvanhisnaef@polije.ac.id

Received: 6 March 2024 – Revised: 30 April 2024 – Accepted: 5 May 2024

ABSTRACT

Diabetic retinopathy is a common complication of diabetes mellitus, leading to damage and blockage of retinal blood vessels. Early and accurate detection of diabetic retinopathy severity levels is crucial for timely treatment and prevention of blindness. Diagnostic methods rely on manual examination and human interpretation, resulting in slower and less efficient treatment processes. As a branch of artificial intelligence, computer vision offers a potential solution to analyze retinal images quickly and accurately. The developed system employs image processing techniques and a CNN-based classification model to detect and classify the severity levels of diabetic retinopathy. By providing an automated and efficient approach, the system aims to assist doctors and optometrists in making informed decisions and reducing subjectivity in diagnosis. Early detection through this system can facilitate prompt treatment and improve patient outcomes. The developed system achieves promising results through experimentation and testing with various datasets, with accuracy ranging from 80% to 97%. This project's integration of artificial intelligence, machine learning, and image processing technologies demonstrates their potential in healthcare applications, particularly in diabetic retinopathy diagnosis.

Keywords*: convolution neural network, DenseNet, diabetic retinopathy, image processing, Keras, Tensorflow*.

I. INTRODUCTION

DIABETIC retinopathy is a compilation of diabetes mellitus characterized by damage and blockage of the retinal blood vessels. Early symptoms of diabetic retinopathy are the formation of microaneurysms and leaks in blood vessels, venous abnormalities, retina swelling, abnormal growth of new blood vessels, and damage to nerve tissue [1]. There are two main stages of diabetic retinopathy: Non-Proliferative Diabetic Retinopathy (NPDR) and Proliferative Diabetic Retinopathy (PDR). The NDPR stage is divided into three stages: mild, moderate, and severe [2]. Diabetic retinopathy at all stages can cause decreased vision within six months, accompanied by the appearance of black shadows in the vision. It can even cause blindness if not treated quickly [3].

Early and accurate diagnosis of diabetic retinopathy is very important to minimize blindness and improve medical care. Ophthalmologists recommend that diabetic patients be screened at least twice a year to immediately diagnose signs of diabetic retinopathy [4]. Currently, the treatment of diabetic retinopathy is carried out through several examinations, including biomicroscopy, fluorescent angiography, ultrasonography, and Optical Coherence Tomography (OCT) [5]–[7]. The diagnosis is carried out through direct observation using a fundus camera and requires expert human interpretation, so the treatment process for patients becomes slower and inefficient [8]. A quick and precise examination aims to carry out stages of retinal therapy in patients through photocoagulation laser therapy and vitrectomy [9]. Computer Vision and image processing is the right solution to analyze retinal images in diabetic retinopathy patients quickly [10].

Computer vision in the modern era is widely applied in various fields such as agriculture, industry, and especially in the health sector. Computer vision involves developing computer algorithms to simulate human visualization and extract information from objects [11]. It is a branch of artificial

intelligence related to human vision. Visually, the human eye and computer vision systems function similarly. The goal is to interpret spatial data, which is data indexed by more than one dimension. In this study, researchers applied the Convolutional Neural Network (CNN) as a classification method to build a detection model.

The purpose of early detection of diabetic retinopathy in this study is to build a web-based system that can classify and detect the severity level of diabetic retinopathy from images of the eye's retina. This system is intended to help doctors and optometrists make decisions and reduce the subjectivity involved in diagnosing diabetic retinopathy. Additionally, it is expected to facilitate early detection, enabling timely and efficient treatment.

II. RESEARCH METHOD

Diabetic retinopathy is a disease affecting individuals with diabetes mellitus, often initially presenting without symptoms. Patients with diabetes mellitus for 5-15 years have a 40-50% chance of developing diabetic retinopathy, increasing to 60% after 15 years [12]. Diabetic retinopathy is characterized by blood vessel disorders in the retina, such as leakage and blockage, which can lead to the formation of fragile, abnormal blood vessels and cause bleeding in the eye [13].

Currently, the diagnosis of diabetic retinopathy is typically performed manually by observing retinal images taken with a fundus camera. Ophthalmologists usually advise diabetic patients to be screened at least twice a year for signs of diabetic retinopathy [4]. Several methods are used to diagnose diabetic retinopathy, including biomicroscopy, fluorescein angiography, ultrasonography, and Optical Coherence Tomography (OCT) [5]–[7]. Another common method involves observing fundus images, such as assessing the retina's orange color and the optic disc's edge and color [14]. Diagnosing in this manner requires a high level of concentration, making the treatment process slower and inefficient [15].

A. Convolution Neural Network (CNN)

A Convolutional Neural Network (CNN) is an algorithm that recognizes images by convolving different kernels with them to extract various features. These features are processed in different layers called neural layers or fully connected layers, which mirror the biological neural network of the brain. The key principle is to take the weighted contribution of each feature to identify images that share the effects of all features [16]. Convolutional layers use specific kernels convolved with the image to recognize desired patterns, colors, or shapes, including the initial and intermediate layers of the CNN. The initial layers of a CNN are convolutional layers that extract visible features from an image. The next layers process the output of the initial layers by convolving with kernels to extract more abstract and hidden features. The final layer uses a neural network to process the weighted influence of all features to recognize the image [17]. Despite the many advantages of CNNs, fully training a CNN from scratch requires large computational resources [18]. This can be mitigated with large image datasets, such as ImageNet, and pre-trained models provided by frameworks like Keras [19].

B. TensorFlow and Keras

TensorFlow is an open-source machine learning framework developed by Google. TensorFlow is a powerful and flexible machine learning framework that allows developers to build a wide variety of machine learning models, including deep neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) [20]. It provides a variety of tools and libraries to simplify the process of building and training these models, including data preprocessing, model optimization, and deployment.

Keras, on the other hand, is a high-level neural network API written in Python and running on top of TensorFlow. It provides an easy-to-use interface for building deep learning models, enabling developers to easily design and train complex neural networks with just a few lines of code [21].

Keras is designed to reduce the cognitive load on end users so they can focus on building models. As such, Keras is great for beginners; for many, it is the entry point into machine learning. At the same time, Keras seamlessly integrates with its TensorFlow backend, allowing users to create any model they can implement in pure TensorFlow [22]. This flexibility makes Keras an excellent tool for even experienced deep learning practitioners.

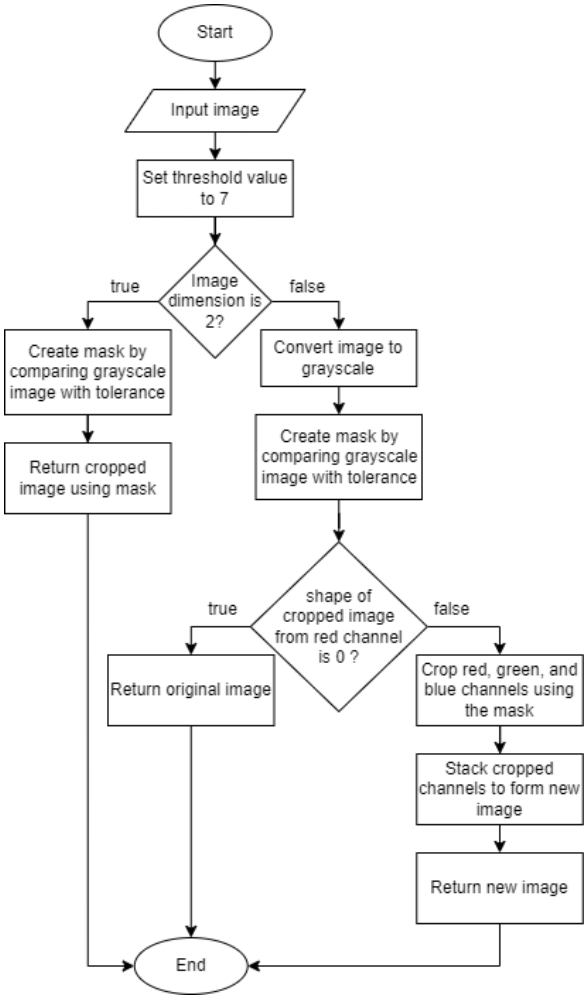

Figure 1. Background Cropping

Keras provides Keras Applications, which are deep learning models used together with pre-trained weights. These models can be used to predict and perform feature extraction or fine-tuning. These models include Xception, VGG16, VGG19, ResNet50, InceptionV3, InceptionResNetV2, MobileNet, DenseNet & NasNet [17], [19].

C. DenseNet

DenseNet (Dense Convolutional Network) is a deep learning architecture, one of the pre-trained models supported by the Keras Application developed by Huang G [23]. DenseNet is based on densely connected layers, where each layer is connected to every other layer in a feed-forward fashion. This architecture supports feature reuse and encourages feature propagation throughout the network, leading to improved performance and more efficient use of parameters.

In DenseNet, each layer inputs all feature maps from all previous layers and concatenates them along the channel dimension. The resulting feature maps are then passed through dense layers (convolutional layers followed by batch normalization layers and non-linear activation functions) to generate new feature maps [23]. This process is repeated for each layer in the network.

DenseNet architecture consists of several different models, each with a different number of layers and parameters. The models are DenseNet-121, DenseNet-169, DenseNet-201, and DenseNet-264 [24]. DenseNet has achieved state-of-the-art performance on various image classification tasks, including the ImageNet dataset, and has also been applied to other computer vision tasks such as object detection and segmentation.

D. Preprocessing Image with Background Cropping

Based on Figure 1, the flowchart begins with the input image and sets the threshold value to 7. The first decision block checks if the input image has a dimension of 2. If the answer is "yes," the control

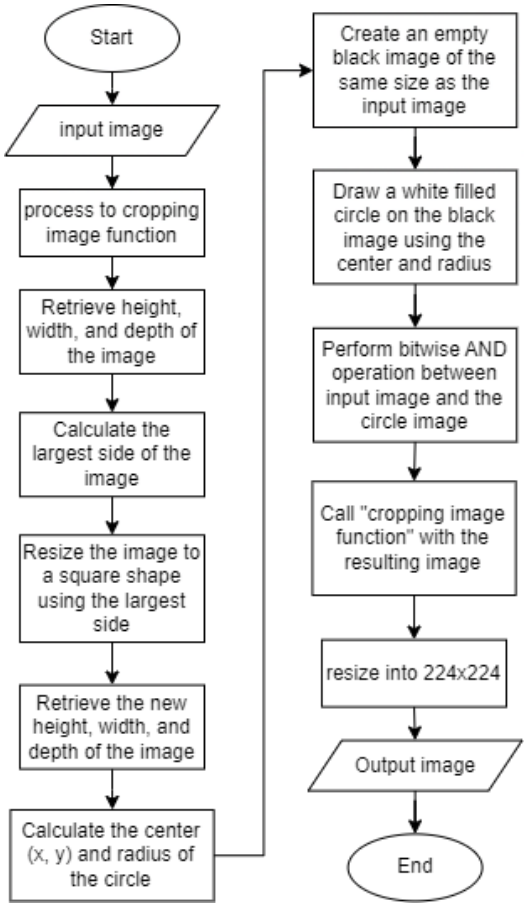

Figure 2. Circle Crop Image

flow proceeds to the left, creating a mask by comparing the image with a tolerance value. This mask is then used to crop the image, and the cropped image is returned. If the answer is "no," the control flow proceeds to the right.

E. Preprocessing Image with Circle Crop Image

Based on Figure 2, the main function of the circle crop is to crop the retinal image, which has a non-circular shape, into a circular retina image. The circle crop process starts by cropping the unnecessary background. Next, the largest side of the height and width is found, and the image is resized to that size. The following step is to compute the image's center coordinate by dividing the image's width and height by 2 and converting the result to an integer. Then, the radius of the circle that will be used as a mask on the image is calculated. The next step involves creating a black image with the same di-mensions as the original image and drawing a white circle. Afterward, the circular mask is applied to the original image. The final step is to crop unnecessary backgrounds and resize the image to 224x224 pixels for convolution and classification.

F. Classification using Convolution Neural Network

This classification system utilizes the Convolutional Neural Network (CNN) method and Dense-Net pre-trained model to classify five severity levels of diabetic retinopathy: normal, mild NPDR, moderate NPDR, severe NPDR, and PDR. The classification process is illustrated in Figure 3.

The first step is to load the image data for training and validating the model. The next step in-volves cropping the image to a circle and resizing it to 224x224 pixels to ensure consistency in size and shape. Following this, classification labels for the images are created. In a multilabel classifica-tion problem, each retina image has multiple labels. For example, encoding a class PDR diabetic ret-inopathy will predict [1, 1, 1, 1, 1].

The dataset is then divided into two sets: a training set for training the model and a validation set for evaluating the model's performance during training. A data generator is created to generate batches of

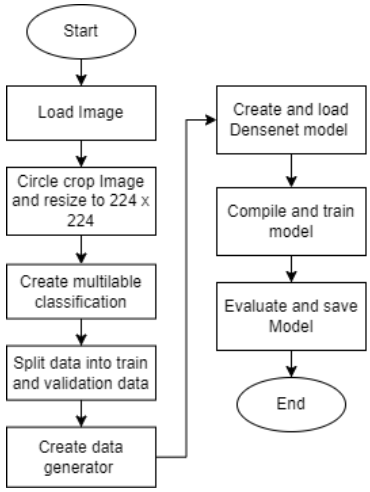

Figure 3. Classification using Convolution Neural Network

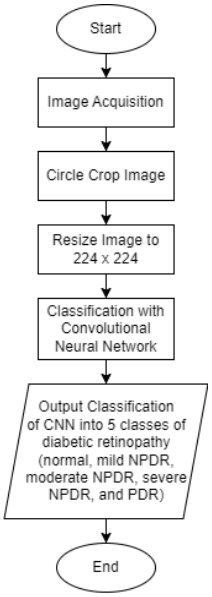

Figure 4. Classification using Convolution Neural Network

training and validation data, performing random transformations on the datasets, such as vertical/horizontal flipping, rotation, and zooming. Next, the DenseNet pre-trained model is loaded to build the neural network model. The model is then compiled, trained, evaluated, and saved for detection.

G. System Flowchart

Based on Figure 4, the system flowchart outlines the system's construction, beginning with image acquisition to ob-tain the retina's image data. The image pre-processing stage involves cropping images into circles to remove irrelevant parts and resizing them to 224x224 pixels to maintain consistent size and shape, which is crucial for training a deep learning model. The final step employs a convolutional neural network (CNN) to classify the retinal images into one of five classes of diabetic retinopathy: normal, mild non-proliferative diabetic retinopathy (NPDR), moderate NPDR, severe NPDR, and proliferative diabetic retinopathy (PDR).

III. RESULT AND DISCUSSION

A. Image Processing

This project was trained and validated with a dataset containing 3112 training data and 550 validation data divided into five severity classes of diabetic retinopathy. The classification process starts with image processing to a machine learning model that can detect five severity levels of diabetic retinopathy automatically. The first retinal image classification process is image processing by cutting off unused

TABLE 1
MULTICLASS TO MULTI-LABEL CLASSIFICATION

Diagnosis	Class	Array	Multi-label	Multi-label Array
Normal	0	[1,0,0,0,0]	0	[1,0,0,0,0]
Mild Non-Proliferative Diabetic Retinopathy (NPDR)	1	[0,1,0,0,0]	0,1	[1,1,0,0,0]
Moderate NPDR	2	[0,0,1,0,0]	0,1,2	[1,1,1,0,0]
Severe NPDR	3	[0,0,0,1,0]	0,1,2,3	[1,1,1,1,0]
Proliferative Diabetic Retinopathy (PDR)	4	[0,0,0,0,1]	0,1,2,3,4	[1,1,1,1,1]

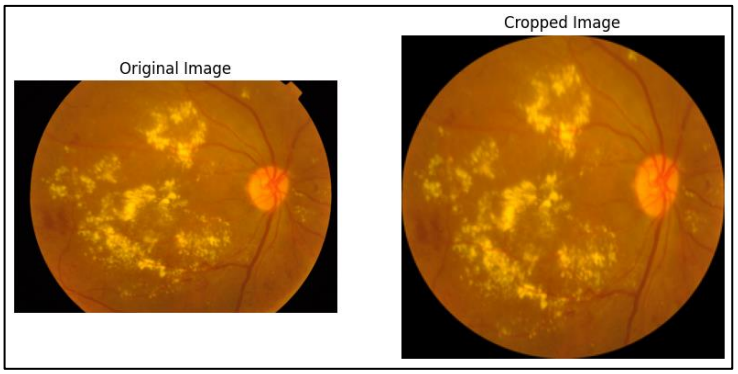

Figure 5. Example of image processing result

backgrounds in retinal images and equalizing the image dimensions. The function of image processing is that the classification results obtain high accuracy. Figure 5 is an example of an image processing result.

B. Classification with Convolution Neural Network

1) Creating multilabel classification

The first step of classification is transforming a binary classification problem into a multi-label classification problem. It initializes a new array called y_train_multi with the same shape as the original y_train array. It starts by copying the labels from the last column of y_train into the correspond-ing column of y_train_multi. Then, it iterates through the remaining columns in reverse order, per-forming a logical OR operation between the current column and the labels of the next column and assigning the result to the current column of y_train_multi. This process effectively combines the labels for each class, creating a multi-label representation of the target variable.

For example, the Proliferative Diabetic Retinopathy (PDR) class which represents class 4 will be converted into an array [0,0,0,0,1] using pandas library. From this array, it will be transformed into a multi-label classification [1,1,1,1,1]. Table 1 is the details of the multi-label classification transformation for each class.

2) Creating data generator and metric callback

The image data generator is configured with various augmentation techniques, such as zooming, flipping, and filling modes. This data generator will produce augmented batches of training data on-the-fly during model training, enhancing the model's ability to generalize and handle variations in the input data. Additionally, a custom metric class is implemented to calculate and store evaluation metrics, such as classification reports and confusion matrices, at the end of each training epoch. It uses the training model's validation data and predicted labels to compute these metrics using functions from the scikit-learn library. The computed metrics are stored in lists for later analysis and reporting.

3) Building a Convolution Neural Network and training the model

The next step involves creating a model based on the DenseNet121 architecture. The DenseNet model is loaded with pre-trained weights, excluding the fully connected layers. The model is then built by adding additional layers, such as global average pooling, dropout, and a dense layer. The model is compiled with a binary cross-entropy loss function, Adam optimizer, and accuracy as the evaluation metric. After the model is built, it is trained using the fit_generator method with augmented batches of training data. This method specifies the number of steps per epoch, the number of epochs, the validation data, and a callback function (custom_metrics) to calculate and store evaluation metrics. After training, the model is saved to a file for future use.

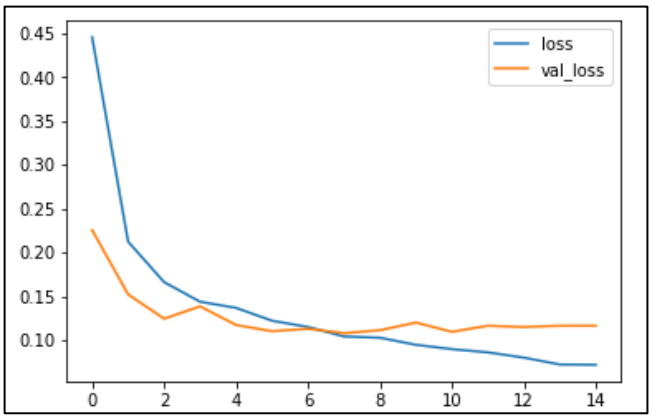

Figure 6. Training and validation loss values

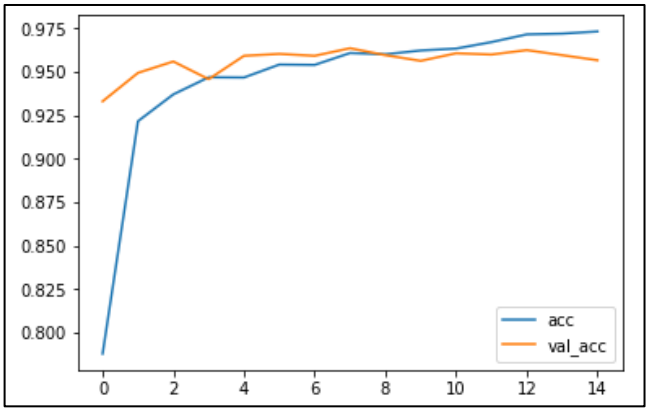

Figure 7. Training and validation accuracy values

4) Evaluating training history

The last step is creating a data frame from the training history and plots the training and validation loss values and the training and validation accuracy values. These plots can help visualize the model's training progress and identify overfitting or underfitting issues. Figures 6 and 7 plot the machine learning model's training history. Figure 6 shows the training and validation loss values, which decrease in each epoch. Figure 7 shows the training and validation accuracy values, which increase in each epoch.

In Figure 7, the first epoch of the model achieves a decent training accuracy of around 78%, indicating that the model is learning patterns from the data. The validation accuracy is much higher at 92.30%, showing that the model can generalize well to unseen data even at the beginning of training. As training progresses, both training and validation accuracies continue to increase, which is a good indication that the model is learning more complex patterns and features from the data over time, resulting in improved performance. In Figure 6, the validation loss consistently decreases throughout the training period, indicating that the model is becoming more confident in its predictions on unseen data. This is an important metric as it shows how well the model generalizes and avoids overfitting. Around epoch 10, the increase in training and validation accuracy becomes slight, which is common and indicates that the model may have reached its limit in learning from the current data architecture and setup.

It is important to pay attention to the validation metrics, as they indicate how well the model will perform on new, unseen data. In medical applications like diabetic retinopathy classification, high validation accuracy is crucial for reliable predictions on patient data. One potential obstacle could be the limited size or diversity of the training data. If the dataset is not representative of the full spectrum of diabetic retinopathy cases, the model might struggle to generalize well. Data augmentation techniques and ensuring a diverse dataset can help mitigate this issue.

C. Evaluating the Machine Learning Model

The machine learning evaluation process uses several datasets from Kaggle competitions, including the Aptos 2019 Blindness Detection, EyePACS, and the Indian Diabetic Retinopathy Image Dataset. Table 2 provides a detailed summary of the machine learning evaluation results.

TABLE 2
EVALUATION OF MACHINE LEARNING MODEL

Dataset	Total Data	Test Accuracy
Aptos 2019 Blindness Detection	550	97%
EyePACS	35108	85%
Indian Diabetic Retinopathy Image Dataset	413	83%

The decrease in accuracy when testing on different datasets suggests that the model's performance may not generalize well across domains. Each dataset may have unique characteristics, such as image quality, patient demographics, and disease severity distribution, which can affect model performance. Despite the decrease in accuracy with the new datasets, achieving 85% accuracy on the EyePACS dataset and 83% on the Indian Diabetic Retinopathy Image Dataset still indicates a reasonable level of performance. Understanding the limitations of model generalization and addressing them can lead to improved performance across different datasets.

IV. CONCLUSION

This project presents a method for detecting diabetic retinopathy severity levels using a convolutional neural network. It comprises two main steps: image processing and classification using a convolutional neural network to detect five severity levels of diabetic retinopathy. The project proposes cropping and resizing processes in image processing and model training processes in convolutional neural networks. The proposed system has been tested using several datasets and achieves 80% to 97% accuracy. To further improve the model, experimenting with different architectures (e.g., deeper networks, different types of layers), hyperparameter tuning (learning rate, batch size), transfer learning, and data preprocessing techniques (normalization, augmentation) could enhance overall performance and generalization.

ACKNOWLEDGMENT

I would like to thank my supervisors for their unwavering support throughout the project. I had a great time working with both of them and am deeply grateful for their guidance. I also extend my gratitude to my family and friends for their constant support. This project would not have reached this stage without their encouragement and assistance.

REFERENCES

[1] A. B. Wiratama, Y. Fu'adah, S. Saidah, R. Magdalena, I. D. S. Ubaidah, and R. B. J. Simanjuntak, "Diabetic Retinopathy Classification Based on Fundus Image Using Convolutional Neural Network (CNN) with MobilenetV2," in *Proceeding of the 3rd International Conference on Electronics, Biomedical Engineering, and Health Informatics: ICEBEHI 2022, 5–6 October, Surabaya, Indonesia*, Springer, 2023, pp. 89–102.

[2] S. Qummar *et al.*, "A deep learning ensemble approach for diabetic retinopathy detection," *Ieee Access*, vol. 7, pp. 150530–150539, 2019.

[3] W. Matuszewski, E. Bandurska-Stankiewicz, R. Modzelewski, U. Kamińska, and M. Stefanowicz-Rutkowska, "Diagnosis and treatment of diabetic retinopathy—historical overview," *Clin. Diabetol.*, vol. 6, no. 5, pp. 182–188, 2017.

[4] A. Pak, A. Ziyaden, K. Tukeshev, A. Jaxylykova, and D. Abdullina, "Comparative analysis of deep learning methods of detection of diabetic retinopathy," *Cogent Eng.*, vol. 7, no. 1, p. 1805144, 2020.

[5] A. Invernizzi, M. Pellegrini, E. Cornish, K. Y. C. Teo, M. Cereda, and J. Chabblani, "Imaging the choroid: from indocyanine green angiography to optical coherence tomography angiography," *Asia-Pacific J. Ophthalmol.*, vol. 9, no. 4, pp. 335–348, 2020.

[6] C. Stathopoulos *et al.*, "Successful conservative treatment of massive choroidal relapse in 2 retinoblastoma patients monitored by ultrasound biomicroscopy and/or spectral domain optic coherence tomography," *Ophthalmic Genet.*, vol. 39, no. 2, pp. 242–246, 2018.

[7] A. S. Vergmann *et al.*, "Optical coherence tomography angiography measured area of retinal neovascularization is predictive of treatment response and progression of disease in patients with proliferative diabetic retinopathy," *Int. J. Retin. Vitr.*, vol. 6, pp. 1–7, 2020.

[8] B. Tymchenko, P. Marchenko, and D. Spodarets, "Deep learning approach to diabetic retinopathy detection," *arXiv Prepr. arXiv2003.02261*, 2020.

[9] J. Lin, J. S. Chang, N. A. Yannuzzi, and W. E. Smiddy, "Cost evaluation of early vitrectomy versus panretinal photocoagulation and intravitreal ranibizumab for proliferative diabetic retinopathy," *Ophthalmology*, vol. 125, no. 9, pp. 1393–1400, 2018.

[10] A. Jamal, M. Hazim Alkawaz, A. Rehman, and T. Saba, "Retinal imaging analysis based on vessel detection," *Microsc. Res. Tech.*, vol. 80, no. 7, pp. 799–811, 2017.

[11] S. Liu and Y. Liu, "Application of human movement and movement scoring technology in computer vision feature in sports training," *IETE J. Res.*, pp. 1–7, 2021.

[12] D. J. Sundoro, R. Patmasari, and R. Magdalena, "Klasifikasi retinopati diabetik non-proliferatif dan proliferatif berdasarkan citra fundus menggunakan metode gabor wavelet dan klasifikasi jaringan saraf tiruan backpropagation," *eProceedings Eng.*, vol. 6, no. 2, 2019.

[13] A. Kwasigroch, B. Jarzembinski, and M. Grochowski, "Deep CNN based decision support system for detection and assessing the stage of diabetic retinopathy," in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, IEEE, 2018, pp. 111–116.

[14] D. U. R. Yani and D. R. Sulistyaningrum, "Klasifikasi Tingkat Keparahan Non-ProliferativeI Diabetic Retinopathy Bedarsarkan Hard Exudate Menggunakan Extreme Learning Machine," *J. Sains dan Seni ITS*, vol. 6, no. 2, pp. A89–A94, 2017.

[15] F. Oktavia, D. Andreswari, and B. Susilo, "Segmentasi Citra Diaretdb1 Pada Area Hemorrhages Diabetic Retinopathy Menggunakan Metode Region Growing," *Rekursif J. Inform.*, vol. 10, no. 1, pp. 1–11, 2022.
[16] C.-H. Lee and Y.-H. Ke, "Fundus images classification for diabetic retinopathy using deep learning," in *Proceedings of the 13th International Conference on Computer Modeling and Simulation*, 2021, pp. 264–270.
[17] S. H. Khan, Z. Abbas, and S. M. D. Rizvi, "Classification of diabetic retinopathy images based on customised CNN architecture," in *2019 Amity International conference on artificial intelligence (AICAI)*, IEEE, 2019, pp. 244–248.
[18] M. A. Morid, A. Borjali, and G. Del Fiol, "A scoping review of transfer learning research on medical image analysis using ImageNet," *Comput. Biol. Med.*, vol. 128, p. 104115, 2021.
[19] M. A. Pangestu and H. Bunyamin, "Analisis Performa dan Pengembangan Sistem Deteksi Ras Anjing pada Gambar dengan Menggunakan Pre-Trained CNN Model," *J. Tek. Inform. dan Sist. Inf.*, vol. 4, no. 2, pp. 341–348, 2018.
[20] C. Francois, "Deep learning with Python." Manning Publications, 2018.
[21] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.
[22] D. Grattarola and C. Alippi, "Graph neural networks in tensorflow and keras with spektral [application notes]," *IEEE Comput. Intell. Mag.*, vol. 16, no. 1, pp. 99–106, 2021.
[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
[24] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Q. Weinberger, "Convolutional networks with dense connectivity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8704–8716, 2019.