Diabetic Retinopathy Detection on Retinal Fundus Images Using Convolutional Neural Network

Adarsh Pradhan(✉), Bhaskarjyoti Sarma, Rahul Kumar Nath, Ajay Das, and Anirudha Chakraborty

Department of Computer Science and Engineering, Girijananda Chowdhury Institute of Management and Technology, Azara, Guwahati, India
adarsh@gimt-guwahati.ac.in,
bhaskarjyotisarma85@gmail.com,
rahulnath847@gmail.com, ajay999das@gmail.com,
imanirudhachakraborty64@gmail.com

Abstract. Diabetic retinopathy is a serious eye disease that occurs due to diabetes mellitus, also commonly known as diabetes, and it has grown as the most common cause of blindness in the present world. It is a disease in which the blood vessels behind the retina are damaged. At first, it shows no symptoms, but with time, it eventually leads to blindness. Early diagnosis of diabetic retinopathy can prevent vision loss in patients. The method proposed here for the detection of diabetic retinopathy disease is based on a convolutional neural network that categorizes the fundus images of patients according to the severity level of diabetic retinopathy. The input images are collected from the Kaggle diabetic retinopathy dataset, and various preprocessing steps such as cropping, resizing, grayscaling, CLAHE and Min-Max normalization are performed. Precision, recall and Kappa score values of our model are highly affected, due to the unequal distribution of dataset.

Keywords: Diabetic retinopathy · Convolutional neural network · Fundus images · CLAHE · Kappa score

1 Introduction

Diabetes, also known as diabetic mellitus (DM), is a growing disease worldwide. The World Health Organization statistics show that the count of people suffering from DM is going to reach 439 million by 2030 [1]. One of the major complications caused by DM is diabetic retinopathy (DR), which is now a major cause of blindness. Diabetic retinopathy is a severe eye disease caused due to high sugar level content in blood, which results in damage caused to the blood vessels behind the retina [2, 3]. This is usually followed by swelling and leakage. At the advanced stage, the retina starts producing tiny new blood vessels that might bleed, causing vision blockage leading to permanent vision loss. The earliest signs of retinal damage are shown by the formation of microaneurysm followed by hard exudates, soft exudates, hemorrhages, macular edema, etc. Several screening tools are used for diabetic retinopathy detection, such as

A. Bhattacharjee et al. (Eds.): MIND 2020, CCIS 1240, pp. 254–266, 2020.
https://doi.org/10.1007/978-981-15-6315-7_21

Optical Coherence Tomography (OCT) or fundus image examination [4]. Here we use fundus images for the detection of Diabetic Retinopathy (DR) stages. We use the Kaggle diabetic retinopathy dataset, which is of size 82 GB and consists of 35,126 RGB retinal images. It consists of several stages of DR as shown from Figs. 1, 2, 3, 4 and 5, where a clinician has rated each image on a scale of 0 to 4, where, 0 signifies No DR (Fig. 1), 1 signifies Mild DR (Fig. 2), 2 signifies Moderate DR (Fig. 3), 3 indicates Severe DR (Fig. 4) and 4 indicates Proliferative DR (Fig. 5). The images are first pre-processed and then fed to the CNN model for training, validation and testing.

Fig. 1. No DR (Stage 1)

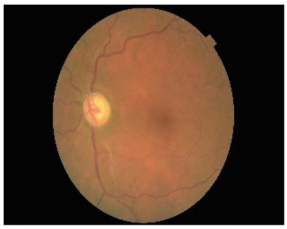

Fig. 2. Mild DR (Stage 2)

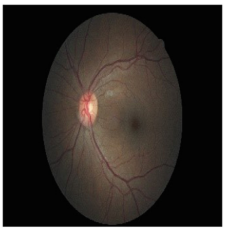

Fig. 3. Moderate DR (Stage 3)

Fig. 4. Severe DR (Stage 4)

Fig. 5. Proliferative DR (Stage 5)

Early detection and regular screening for DM can reduce the effect of DR, but it's not cost-efficient [5]. Also, highly skilled technicians are required for the examination of the fundus image, which is not quite possible, especially in rural areas where the rate of diabetes is high. So, this has led to the need for creating new automated detection tools that can help tackle the problem relating to diabetes. We propose to address the problem using a CNN based method that can train on the fundus images and classify them to the different stages of DR. The proposed CNN architecture consists of 13 layers, as shown in Fig. 12. The first two convolutional layers have 32 filters of size 3 * 3, which are followed by the ReLu layer and the max-pool layer. The third convolutional layer is as same as before, except it has 64 filters of size 3 * 3. The last set consists of a fully connected layer having 64 nodes and ReLu as activation function followed by a dropout layer with a dropout ratio of 0.5, which is ultimately followed by

another fully connected layer of 5 nodes with softmax as an activation function. Finally, precision, recall and Kappa score values are evaluated to examine the performance of our model.

2 Related Work

We have seen that earlier methods were divided into feature extraction and classification category. Methods like SVM, K-NN, etc., are still useful for classification. However, the complexity and computation time are much higher because it depends upon manual procedures [2, 6]. So, to extract features, many researchers are replacing these two-step methods with deep learning techniques [2], where the deep learning method that they have used are based on some architectures like AlexNet, ResNet-50 etc. [2].

Zhiguang and Jianbo Yang [2] proposed the model that includes regression activation mapping which shows the area of the desired region in the fundus image. They have also illustrated that by using the global average pooling layer instead of the fully connected layer in their Net-5 architecture, a good kappa score can be acquired and training speed can be increased.

Xianglong Zeng et al. [7] have developed a system that classifies the fundus images with or without DR which takes fundus images of both eyes as input and calls novel Siamese-like CNN model based on Inception V3 architecture and achieved kappa score of 0.829, AUC of 0.951, sensitivity and specificity of 82.2%, 70.7% respectively. In another proposed method [6], they have used three steps which are pre-processing, retinal segmentation, and image classification. They have used U-Net for retinal segmentation, which significantly improved the output of classifying 5 stages of DR namely - No DR, Mild DR, Moderate NPR, Severe NPR, PDR.

In another paper [8], they have used Inception-V3 architecture, which extracts multilevel features from the fundus images and the model is trained by taking different sizes of datasets giving an accuracy of 67% (for 10,000 fundus images) and 88% (for 40,000 fundus images). Also, based upon different hyperparameters, they have obtained different accuracies; for instance, they have taken different number of training epochs like 100 and 200, where they got the accuracy of 70% and 88%, respectively, and concluded that the accuracy of the model could be increased by training the model for more number of epochs, though after a certain point, further number of epochs did not change the accuracy of the model.

In [16], they have proposed a method to divide the EyePacs datasets into two sections one for left eye and other for right eye, then train them individually using the CNN model. They obtained an accuracy of 63.6% and 66.4% respectively.

In [5], Xiaoliang Wang et al. applied different types of deep neural architectures to classify DR in five stages. They have used the EYEPACS dataset to train 166 fundus images on AlexNET, VGG16, InceptionNet, giving the accuracy of 37.43%, 50.03%, 63.23% respectively, with InceptionNet architecture giving the highest accuracy among the three. In [9], Gen-Min Lin et al. showed that the use of entropy images increased the accuracy of their model by 4.3%, where the accuracy of 81.80% on original fundus images increased to 86.10% for the entropy images.

3 Proposed Method

Figure 6 gives a brief description of the CNN based method that we have employed for the detection of Diabetic Retinopathy (DR) detection.

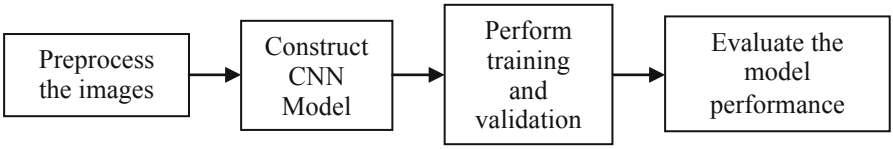

Fig. 6. CNN based method for DR detection

3.1 Preprocessing

In the preprocessing step, we have performed Cropping, Resizing, Grayscaling, Histogram Equalization and Normalization to standardize the high-resolution fundus images as illustrated below.

Cropping. All the fundus images in the dataset are cropped by a percentage in such a way that most of the unwanted black pixels are removed without cutting off the region of interest. Here we have cropped by 10% from the left and the right side and by 0.5% from the top and the bottom side of the original image (Fig. 7) to obtain the cropped image (Fig. 8).

Fig. 7. Original image

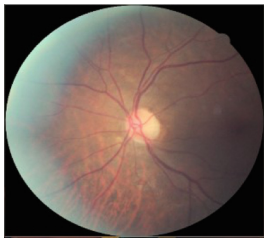

Fig. 8. Cropped image

Resizing. The Kaggle dataset contains high-resolution images and training those images in the convolutional neural network requires a system with a high-end configuration. Therefore, in order to reduce the network's training time, we resized the cropped image to 256 $\times$ 256 using bilinear interpolation method.

Grayscaling the Images. In an image, one color pixel is made with three colors, namely, red, green and blue (RGB), having a total bit size of 24, which is represented by three arrays. In a grayscale image, the grey pixel is represented by only one dimension with a bit size of 8, therefore converting RGB to grayscale helps us to reduce the computation time for the model and also grayscale image takes less space than RGB image. So, after we crop the original image (Fig. 9), we convert it to grayscale (Fig. 10).

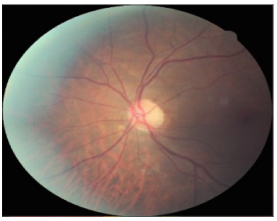

Fig. 9. Cropped image

Fig. 10. Grayscale image

Histogram Equalization and Normalization. Contrast Limited Adaptive Histogram Equalization is performed over ordinary AHE in order to minimize the over-amplification of noise in near contrast regions [10]. Histogram equalization equalizes the contrast level in higher and lower contrast portions of the image and enhances edges. As shown in Fig. 12, CLAHE is applied to the image in Fig. 11. The resized images that are in RGB format are converted to grayscale image, applied CLAHE, and then Min-Max normalization is performed in order to scale the image values within a range of 0 and 1. Using Eq. 1 we can rescale an image's pixel intensities within a range [0,1]:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

where x is the original value and x' is the new normalized value.

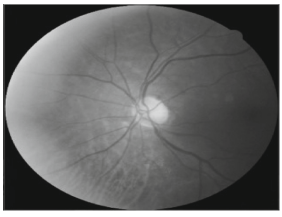

Fig. 11. Grayscale image

Fig. 12. CLAHE image

3.2 Convolutional Neural Network

In recent years, the deep learning algorithms which imitate the operations of the human cerebral cortex have been found of excellent use [11], and one of the vastly used deep learning algorithms for image processing is a convolutional neural network, also known as ConvNet or CNN. CNN can automatically extract local features of an image by applying a set of weights called filters and applied multiple filters to extract different features [2]. There are many CNN architectures like AlexNet (which has 8 layers and can process 61 million parameters), VGG (19 layers), GoogleNet (22 layers and can process 5 million parameters), ResNet (152 layers) and in all these CNN networks they have many hidden layers, and as the layers go deeper, more features are extracted for classification [12].

Convolution Layer. The convolutional layer contains a set of filters that are used to extract features. When these filters are convolved with the input volume, it produces feature maps containing convoluted values, computed by the convolution operation occurring between an array of inputs and filters. A convolution operation is a matrix dot product between an array of the input volume and a set of filters. A convolution layer can have multiple filters in the network, and with these filters, one can extract features from low-level to high-levels in an image during each training cycle [12, 13].

Pooling Layer. The pooling layer is applied to downsample the input volume received from the previous layers. There are different methods that can be used, such as Average pooling, Max pooling and Global pooling. Max pooling is one of the standard techniques where it calculates the maximum value from a small region say n*n of the input image and store it in the output map. This max-pooling process continues until it moves across every region of the input volume to generate a max-pooling map [6, 12, 13].

Fully Connected Layer. Before giving the input to the fully connected layer for classification, the output from the previous layer is flattened [13]. The back-propagation technique used in the fully connected layer is responsible for the precision of the weights, where each node or neuron receives those weights to determine the most appropriate labels. Finally, the output results from each node are used to make the classification decision.

ReLU Layer. ReLU is the commonly used activation function to add non-linear properties to the inputs from the previous convolution layer. Here the Relu function transforms all the weighted sum of inputs to the maximum of either 0 or the input itself, i.e., it changes all the negative activations value to zero [6, 13].

Dropout Layer. Overfitting is the problem where our model learns too many patterns from our training sets such that it fails in classifying new datasets. That's why the dropout layer is used to prevent the model from overfitting, where it randomly drops out some of the hidden nodes during the training process [11], and as a result, this layer provides some improvements in the accuracy of the model [12, 13].

Softmax Layer. This layer uses the softmax activation function to transform all inputs received from the fully connected layer in the range of 0 to 1, where each output from the softmax layer sums up to 1 forming a probability distribution. The softmax layer is

basically used for multiclass classification by calculating probabilities for different classes [14]. The formula for softmax is shown in Eq. 2:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

where i → 1…, K and input vector (z) → (z_1, $z_{2\ldots}$, z_K) ∈ R^K.

3.3 CNN Architecture

The CNN architecture that we have used consists of 13 layers, as shown in Fig. 13. The first two sets of the layer are the convolution layer, where 32 filters convolve across the image to produce 32 feature maps, then the Relu layer is respectively applied to each feature map for non-linearity, and lastly, the max-pool layer is used to reduce the spatial representation of the feature map. In the third sets of layers, we again use a convolutional layer where 64 filters are used, which is again followed by Relu and max-pool layer.

In the last set of layers, it consists of one fully connected layer of 64 nodes, followed by one dropout layer where the dropout ratio is 0.5, and ultimately it is followed by another fully-connected layer with five nodes and one softmax layer which gives us the final classification outputs.

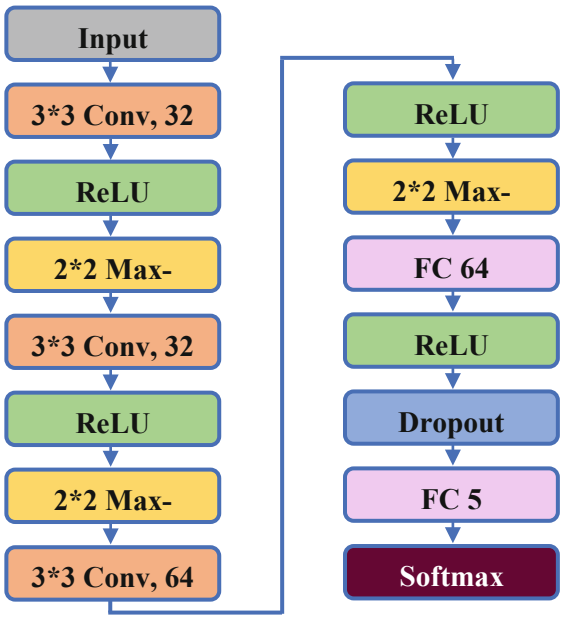

Fig. 13. CNN architecture

3.4 Training Algorithm

We applied Stochastic Gradient Descent with Nesterov Momentum, an optimization algorithm that speeds up training and improves the rate of convergence towards the global minimum of the cost function significantly. Stochastic Gradient Descent differs slightly from regular gradient descent, where it looks for a single mini-batch instead of the whole dataset to calculate the cost and minimize the loss or minimize the global cost function of the network. Nesterov Momentum helps in guiding the gradient always towards the right direction of minimum loss despite the momentum pointing towards the wrong direction as the gradient is not computed from the current position. The rule for updating Stochastic Gradient descent with Nesterov Momentum is shown in Eqs. 3 and 4 [15]:

$$v_t = \mu v_{t-1} - \eta \nabla l(\theta + \mu v_{t-1}) \tag{3}$$

$$\theta_t = \theta_{t-1} + v_t \tag{4}$$

where,

't' → the number of iterations
'μ' → the momentum parameter
'∇' → the gradient
'l' → the loss function
'η' → the learning rate.

3.5 Hyperparameter Tuning

To optimize our model, we have performed tuning of some hyper parameters in the training period. As the performance of any machine learning model is susceptible to its hyperparameters, we, therefore, experimented those by performing training with a small dataset. Having observed the effects, we came up with the values that showed better results, as shown in Table 1. We tuned the initial learning rate to 0.01 with a decay of $10e^{-6}$, which reduces the learning rate after every batch. The batch size was determined in such a way that after every epoch, the learning rate decreases by 10%, as decreasing the learning rate prevents from overshooting the global minimum of the cost function. Also, the momentum for the optimization algorithm was set to 0.9. The update schedule for the learning rate is (Eq. 5):

$$L_t = L_{t-1} * \frac{1}{1 + (D * t)} \tag{5}$$

where,

L → Learning Rate
t → No. of batch iteration
D → Learning Rate Decay

Table 1. Hyperparameters values

Hyperparameters	Values
Initial learning rate	0.01
Learning rate decay	10 e^{-6}
SGD momentum	0.9
Batch size	70
Number of epochs	15

4 Experimental Design and Data

4.1 Dataset

For our proposed model, we have used the EYEPACS dataset, which we collected from the Kaggle website (https://www.kaggle.com/c/diabetic-retinopathy-detection). It contains 35,126 RGB retinal images captured by fundus camera, which are labelled and categorized by the clinicians in five different classes from 0 to 4 depending upon their severity where each class corresponds to different stages of DR as shown above from Figs. 1, 2, 3, 4 and 5. The class distribution of the dataset provided by Kaggle is highly imbalanced which can be seen in Table 2.

Table 2. Class distribution of original datasets

Class	Name	No. of images	Percentage
0	No DR	25810	73.48%
1	Mild DR	2443	6.96%
2	Moderate DR	5292	15.07%
3	Severe DR	873	2.48%
4	Proliferative DR	708	2.01%

4.2 Implementation and Results

Using Open Source Computer Vision (OpenCV) Library, we have performed all the preprocessing steps such as cropping, resizing, greyscaling, CLAHE and Min-Max normalization. After preprocessing, we have divided the whole dataset into 80% as the training set, 15% as the validation set, and the remaining 5% as the testing set. We have fed the training and validation dataset into our model that has been built using Keras API and TensorFlow Library for training the network. In a Windows 10 platform having a system configuration of 16 GB RAM and Intel i7-4790 k 4^{th} generation processor, training of the network has been performed for 15 epochs, as more epochs demand more computational power and high time consumption.

Evaluation Metrics. The accuracy of the model alone cannot evaluate the classifier perfectly, as in this case the distribution of the classes is highly imbalanced. Therefore,

for the evaluation of the performance of our trained model accuracy, confusion matrix, Precision & Recall, and Kappa Score are computed [13].

Accuracy is the score of correctly predicted observations over total observations. The model managed to get a training accuracy of 93.13% but scores about 85.68% accuracy on the testing dataset, which might be due to overfitting as the dataset is highly imbalanced.

The confusion matrix represents the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) values for all the classes. In Fig. 14, a confusion matrix has been shown representing true labels on the vertical co-ordinates and predicted labels on the horizontal co-ordinates.

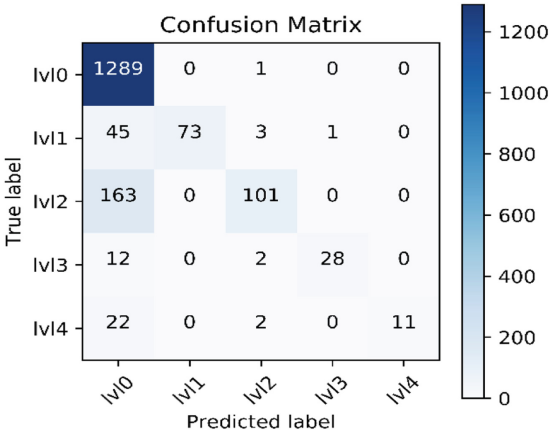

Fig. 14. Confusion matrix

As mentioned above, we have taken 5% of the dataset for testing purpose, shown in Table 3. In the confusion matrix, we see that out of 1290 images of class 0, 1289 images are correctly classified and 1 image is falsely classified. Similarly, for class 1, 73 images are correctly predicted and the rest 49 images are incorrectly predicted.

Table 3. No of testing images

No of testing images	Class label
1290	Class 0
122	Class 1
264	Class 2
43	Class 3
35	Class 4

Using Eqs. 6 and 7, we compute Precision & Recall from the confusion matrix. Precision represents the portion of relevant occurrences among retrieved occurrences, whereas Recall is the portion of the relevant occurrences that are retrieved.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{\mathrm{TP} + \mathrm{FN}} \tag{7}$$

Table 4. Precision and recall

Class label	Recall	Precision
Class 0	0.999	0.841
Class 1	0.598	1.000
Class 2	0.382	0.926
Class 3	0.667	0.965
Class 4	0.314	1.000

In the dataset, the number of instances of class 0 is the highest and the number of instances of class 4 is the lowest, as depicted in Table 3. Because of this highly imbalanced dataset, the classifier has learned more about class 0. This is evident in the precision and recall values shown in Table 4. Since the number of instances of class 0 is the highest, the recall value of class 0 is the highest whereas the precision value is the lowest. Similarly, the number of instances of class 4 is the lowest, so the recall value of class 4 is the lowest, whereas the precision value is the highest.

Cohen's Kappa Score compares the judgment produced by various raters; i.e., how much similarity exists in the judgments. In our case, one of the raters is the ground truth or the true labels, and the other corresponds to our classifier i.e., the predicted labels. Kappa score(K) can be evaluated by Observed Accuracy (p_o) and Expected Accuracy (p_e) as shown in Eq. 8.

$$Kappa\,Score(K) = \frac{p_o - p_e}{1 - p_e} \tag{8}$$

The model gained a Kappa Score of 0.584.

5 Conclusion and Future Scope

In this paper, we use a CNN model to classify various stages of Diabetic Retinopathy (DR) from the fundus images. We use the Kaggle diabetic retinopathy dataset, and perform various preprocessing steps such as cropping, resizing, grayscaling and use CLAHE and Min-Max normalization for further image enhancement. 80% of the

dataset is used for training, 15% for validation and 5% for testing. Multiple metrics are used to evaluate and examine the performance of our model. The Kaggle diabetic retinopathy dataset is highly disproportionate, and it becomes evident by the results of precision and recall values that we acquired. The Kappa Score is also found to be of moderate value. One of our future goals would be to minimize this problem by upscaling the dataset and by using various regularization techniques. Also training in high end system with a greater number of training epochs could further enhance the performance of the network.

We are currently planning to build the Graphical user Interface (GUI) for this project where one can feed the retinal fundus image and have the image classified as DR or non-DR. A detection model of such kind would be highly beneficial to people in rural areas where they may not have instant access to doctors.

Acknowledgments. This work is supported by Assam Science and Technical University, under TEQIP III program of Ministry of Human Resource Development, India, funded by World Bank.

References

1. Kaveeshwar, S.A., Cornwall, J.: The current state of diabetes mellitus in India. Austr. Med. J. **7**(1), 45–48 (2014). https://doi.org/10.4066/AMJ.2013.1979
2. Wang, Z., Yang, J.: Diabetic retinopathy detection via deep convolutional networks for discriminative localization and visual explanation. ArXiv abs/1703.10757 (2017)
3. Wang, X., Lu, Y., Wang, Y., Chen, W.: Diabetic retinopathy stage classification using convolutional neural networks. In: IEEE International Conference on Information Reuse and Integration (IRI), pp. 465–471 (2018). https://doi.org/10.1109/IRI.2018.00074
4. Mookiah, M.R.K., Acharya, U.R., Chua, C.K., Lim, C.M., Ng, E.Y.K., Laude, A.: Computer-aided diagnosis of diabetic retinopathy: a review. Comput. Biol. Med. **43**(12), 2136–2155 (2013). https://doi.org/10.1016/j.compbiomed.2013.10.007
5. Williams, R., Airey, M., Baxter, H., et al.: Epidemiology of diabetic retinopathy and macular oedema: a systematic review. Eye **18**, 963–983 (2004). https://doi.org/10.1038/sj.eye.6701476
6. Burewar, S., Gonde, A.B., Vipparthi, S.K.: Diabetic retinopathy detection by retinal segmentation with region merging using CNN. In: IEEE 13th International Conference on Industrial and Information Systems (ICIIS), pp. 136–142 (2018). https://doi.org/10.1109/ICIINFS.2018.8721315
7. Zeng, X., Chen, H., Luo, Y., Ye, W.: Automated diabetic retinopathy detection based on binocular siamese-like convolutional neural network. IEEE Access **7**, 30744–30753 (2019). https://doi.org/10.1109/ACCESS.2019.2903171
8. Kanungo, Y.S., Srinivasan, B., Choudhary, S.: Detecting diabetic retinopathy using deep learning. In: 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), pp. 801–804 (2017). https://doi.org/10.1109/RTEICT.2017.8256708
9. Lin, G.-M., et al.: Transforming retinal photographs to entropy images in deep learning to improve automated detection for diabetic retinopathy. J. Ophthalmol. (2018). https://doi.org/10.1155/2018/2159702

10. Ma, J., Fan, X., Yang, S.X., Zhang, X., Zhu, X.: Contrast limited adaptive histogram equalization based fusion for underwater image enhancement. Int. J. Pattern Recogn. Artif. Intell. **32**(07) (2018). https://doi.org/10.1142/S0218001418540186
11. Chauhan, R., Ghanshala, K.K., Joshi, R.C.: Convolutional neural network (CNN) for image detection and recognition. In: First International Conference on Secure Cyber Computing and Communication (ICSCCC), pp. 278–282 (2018). https://doi.org/10.1109/ICSCCC.2018.8703316
12. Doshi, D., Shenoy, A., Sidhpura, D., Gharpure, P.: Diabetic retinopathy detection using deep convolutional neural networks. In: International Conference on Computing, Analytics and Security Trends (CAST), pp. 261–266 (2016). https://doi.org/10.1109/CAST.2016.7914977
13. Ghosh, R., Ghosh, K., Maitra, S.: Automatic detection and classification of diabetic retinopathy stages using CNN. In: 4th International Conference on Signal Processing and Integrated Networks (SPIN), pp. 550–554 (2017). https://doi.org/10.1109/SPIN.2017.8050011
14. Kouretas, I., Paliouras, V.: Simplified hardware implementation of the softmax activation function. In: 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), pp. 1–4 (2019). https://doi.org/10.1109/MOCAST.2019.8741677
15. Ruder, S.: An overview of gradient descent optimization algorithms. ArXiv, abs/1609.04747 (2016)
16. García, G., Gallardo, J., Mauricio, A., López, J., Del Carpio, C.: Detection of diabetic retinopathy based on a convolutional neural network using retinal fundus images. In: Lintas, A., Rovetta, S., Verschure, P., Villa, A. (eds.) ICANN 2017. LNCS, vol. 10614. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68612-7_72