https://doi.org/10.46335/IJIES.2023.8.4.1 **e-ISSN: 2456-3463**
Vol. 8 , No. 4, 2023, PP. 1-7

International Journal of Innovations in Engineering and Science, www.ijies.net

Detection of Diabetic Retinopathy Using Deep Learning

Shubham Vartak[1], Ajinkya Parte[2], Soham Ajgaonkar[3], Hitesh Ghagave[4]

[1,2,3,4]UG student, Shree L.R. Tiwari College of Engineering, Thane, India, 401107.

soham.ajgaonkar@slrtce.in

***Received on**: 29 March,2023* ***Revised on**: 18 April,2023* ***Published on**: 20 April,2023*

Abstract— *Diabetic retinopathy is a major problem worldwide and many people are losing their vision because of it. The disease gets severe if it is not treated properly at its early stages. In this disease, the retinal blood vessel gets damaged due to high blood sugar levels which eventually blocks the light that passes through the optical nerves, making the patient with Diabetic Retinopathy blind. Diabetic Retinopathy is detected using manual screening, but this requires a skilled ophthalmologist which may not be available everywhere and thus diagnosis takes a lot of time. Therefore, we decided to build a deep learning model using which we will be able to detect multiple stages of severity for Diabetic Retinopathy. So, we studied and built widely-discussed models - Support Vector Machine (SVM) and Convolutional Neural Network (CNN) - and conducted a comparative study to determine the most suitable model. We found that CNN outperformed SVM in terms of accuracy and efficiency, making it the most suitable model for detecting multiple stages of severity for Diabetic Retinopathy. Thus, this automatic diabetic retinopathy detection model built using CNN can replace manual screening, enabling ophthalmologists to focus on patient care. Additionally, this model can assist inexperienced ophthalmologists in accurately diagnosing diabetic retinopathy.*

Keywords— Diabetic Retinopathy, Blind, Retina, Deep Learning, Ophthalmologist, Manual Screening

I. INTRODUCTION

According to International Diabetes Federation's report, in 2021 approximately 537 million adults (20-79 years) have diabetes. The total number of people having diabetes is projected to rise to 643 million by 2030 and 783 million by 2045. The International Diabetes Federation (IDF) reported that diabetes caused 6.7 million deaths in 2021. In India, 1 in 12 adults, or more than 74 million people living are diabetes patients. So, we are the second highest in the world after China, which has 141 million diabetes patients.

Diabetes is caused because of the excess growth of glucose in the blood. A person affected with diabetes is vulnerable to kidney failure, bleeding teeth, lower limb confiscation, nerve failures, and so on. Having diabetics for a long time causes severe blood vessel damage. Due to this, neurons present in the brain get damaged and cause Diabetic Retinopathy (DR) which results in retinal infection and eventually vision loss.

This is Evidence that DR is a major issue nowadays. This motivated us to build a model which will help to reduce the severity of this problem.

There are various techniques for detecting the affected eye, some of them are slit-lamp biomicroscope, optical coherence tomography (OCT), and fundus images.[5] Symptoms like venous beading, microaneurysms, hemorrhage, neovascularization, etc. are used to diagnose DR [1].

DR is classified into non-proliferative DR (NPDR) and Proliferative DR (PDR). Based on the severity it is further classified as follows:

A. *No apparent retinopathy:* In this stage, there are no signs of diabetic retinopathy in the eye.

B. *Mild non-proliferative retinopathy:* In this stage, there are small areas of balloon-like swelling in the retina's blood vessels. Microaneurysms, Venous beading, Cotton wool spots may be visible.

C. *Moderate non-proliferative retinopathy:* In this stage, some blood vessels that feed the retina are blocked.

https://doi.org/10.46335/IJIES.2023.8.4.1 e-ISSN: 2456-3463
Vol. 8 , No. 4, 2023, PP. 1-7

Hard exudates and Haemorrhages may be visible.

D. *Severe non-proliferative retinopathy:* In this stage, many blood vessels that feed the retina are blocked, leading to the growth of new blood vessels. All the above mentioned signs become more pronounced and Ischemia may be visible.

E. *Proliferative diabetic retinopathy:* In this stage, the new blood vessels are fragile and can bleed into the jelly-like substance in the center of the eye, leading to severe vision loss and even blindness. Neovascularization, Scarring in the retina due to blood vessel damage, Severe haemorrhages, Macular edema, Retinal detachment, Vitreous haemorrhage, Optic neuropathy may be visible.[4]

In many rural areas, there are very few or no experienced ophthalmologists and this delays the diagnosis as well as treatment.[6]

Thus, we decided to build a deep-learning model for diagnosing DR using fundus images.

II. PROPOSED WORK

In this study, we aim to detect diabetic retinopathy using deep learning techniques on fundus images. We first explored the use of SVM (Support Vector Machine), which has been widely used for this purpose.[1] However, after reviewing several recent research papers [3,2], we decided to implement CNN (Convolutional Neural Network) as well, as it is considered to be a better option for this task. In this proposed work, we plan to train and compare the performance of both SVM and CNN models on a large dataset of fundus images. The results of our comparative study will provide insights into the effectiveness of these models for diabetic retinopathy detection.

III. STUDYING DIFFERENT MODELS

There are many models used for Diabetic Retinopathy detection but the most discussed among them were SVM and CNN.

A. *Support Vector Machine*

Diabetic retinopathy is a condition that affects the blood vessels in the retina and can cause vision loss if not detected and treated early. Support vector machine (SVM) is a type of supervised machine learning algorithm that can be used for diabetic retinopathy detection.

The SVM algorithm works by finding a hyperplane in the feature space that separates the different classes of data.

In the case of diabetic retinopathy detection, the features could be extracted from retinal fundus images, such as the presence of microaneurysms, hemorrhages, exudates, and other lesions.

Here are the steps involved in using SVM for diabetic retinopathy detection:

1) Data Collection: Collect a dataset of retinal fundus images that are labeled for diabetic retinopathy. This dataset should include images from both healthy and diabetic patients.

2) Preprocessing: Preprocess the images to remove noise and standardize the image size and orientation. Image normalization techniques can also be used to improve the accuracy of the SVM algorithm.

3) Training: Split the dataset into a training set and a validation set. Train the SVM algorithm on the training set using the extracted features and their corresponding labels.

4) Testing: Test the SVM algorithm on the validation set and measure its performance using evaluation metrics such as accuracy, precision, recall, and F1 score.

5) Deployment: Deploy the trained SVM algorithm on new retinal fundus images to detect diabetic retinopathy.[7]

SVM is a popular machine learning algorithm for diabetic retinopathy detection because it can handle high-dimensional data and is robust to overfitting. However, it requires careful feature selection and hyperparameter tuning to achieve high accuracy.

B. *Convolutional Neural Network:*

Convolutional Neural Networks (CNN) are a type of deep learning algorithm that can be used for diabetic retinopathy detection.

The CNN algorithm works by learning hierarchical representations of the input data through convolutional and pooling layers. In the case of diabetic retinopathy detection, the input data could be retinal fundus images, and the CNN could learn to detect the presence of microaneurysms, hemorrhages, exudates, and other lesions.

Here are the steps involved in using CNN for diabetic retinopathy detection:

1) Data Collection: Collect a dataset of retinal fundus images that are labeled for diabetic retinopathy. This dataset should include images from both healthy and diabetic patients.

https://doi.org/10.46335/IJIES.2023.8.4.1 **e-ISSN: 2456-3463**
Vol. 8 , No. 4, 2023, PP. 1-7

2) Preprocessing: Preprocess the images to remove noise and standardize the image size and orientation. It also includes applying different filters to improve the quality of images. One well-known method of preprocessing is Ben Graham's pre-processing method. It includes resizing the image, Green Channel Extraction, Gaussian Smoothing, Contrast Limited Adaptive Histogram Equalization (CLAHE), and Grey Scaling. It improves the performance of the model.

3) Training: Split the dataset into a training set, validation set, and test set. Train the CNN algorithm on the training set using the preprocessed images and their corresponding labels.

4) Validation: Validate the CNN algorithm on the validation set and measure its performance using evaluation metrics such as accuracy, precision, recall, and F1 score.

5) Optimization: Tune the hyperparameters of the CNN algorithm, such as the number of layers, the size of the filters, the learning rate, and the regularization, to improve its performance on the validation and test sets.

6) Testing: Test the CNN algorithm on the test set and measure its performance using the same evaluation metrics used during validation.

7) Deployment: Deploy the trained CNN algorithm on new retinal fundus images to detect diabetic retinopathy.

CNN is a powerful deep learning algorithm for diabetic retinopathy detection because it can learn complex features from the input data and achieve high accuracy with large datasets. However, it requires a large amount of labeled data and significant computational resources for training and optimization.

IV. Implementation

A. SVM Implementation:

Following are the SVM implementation steps :

1) Importing Data:- We used a dataset consisting of 89 images with corresponding labels indicating affected and unaffected retina. The dataset is divided into two classes where 1 denotes the affected eye, and 0 denotes the unaffected eye.

2) Pre- Processing:- After loading the images we convert them to grayscale using the OpenCV library. After grayscale conversion, adaptive histogram equalization is applied to the image to enhance its contrast. The resulting image is then stored as a flattened 1D numpy array.

This process is repeated for all 90 images in the dataset.

Then the preprocessed grayscale images are loaded and converted to 2D format from their flattened form.

After that, a Discrete Wavelet Transform (DWT) is performed on each image using the 'Haar wavelet' which is one of the simplest and most commonly used wavelets.

Wavelet analysis is a powerful signal processing technique that allows a signal or image to be decomposed into a set of basis functions called wavelets. These basis functions are used to represent the original signal or image in a compressed form, which can be used for various applications such as image compression, noise reduction, and feature extraction.

The Haar wavelet is a type of wavelet function that has a simple step-like shape, making it easy to implement and understand.

The DWT is performed on the 2D preprocessed image. The 2D DWT coefficients obtained from the DWT are then converted into a 1D array by scanning the image in a "snake-like" manner, where the coefficients are read row by row from left to right in odd rows and from right to left in even rows. The resulting 1D array (1D DWT coefficients) serves as a compressed representation of the 2D matrix, while still retaining most of the important information.

This 1D array is then used to create a matched filter bank, which consisted of a set of Gabor filters oriented at different angles. The filters were applied to the 1D DWT coefficients, and the resulting filtered images were flattened into 1D arrays.

The idea behind using the DWT and Gabor filters was to extract features from the input image that would be useful for distinguishing between the different classes.

3) Using K- means clustering:- Next, k-means clustering was applied to each flattened filtered image. The number of clusters was set to 2, which means that each pixel in the filtered image was assigned to one of two clusters. This process produced a set of 1D arrays containing the labels for each pixel in the filtered image.

By using the k-means clustering algorithm, the features were reduced to a set of discrete labels that could be used as input to a classification model. This approach helps to improve the accuracy and efficiency of our machine learning model.

https://doi.org/10.46335/IJIES.2023.8.4.1 **e-ISSN: 2456-3463**
Vol. 8 , No. 4, 2023, PP. 1-7

4) Model training: Finally, the labels for each pixel in each filtered image were concatenated into a single 1D array, which was then used as the input along with class labels corresponding to the input data to train the SVM.

5) Results: The accuracy of the SVM classifier is evaluated using the 'accuracy_score' function from the scikit-learn library. The accuracy of the model is found to be 0.966 i.e approximately 96%, indicating that the model performs well in predicting whether the eye is affected or not.

B. CNN Implementation:

Following are the steps we implemented for CNN implementation:

1) Data Collection:- We used a labeled data set that was freely available on Kaggle. It has 5590 images out of which 1928 are testing images and 3662 are training images. The total dataset size is 10.22GB. The data set also has a CSV file which maps the images to the level of DR. Training Dataset is Balanced.[1]

2) Data Preprocessing:- The data in the dataset was already in the appropriate form for training. So we just resized the images to 300 x 300 pixels.

3) Model Building:

a) Importing libraries and modules:- 1st we imported all necessary libraries and modules like: TensorFlow, Keras, Scikit learn toolkit, TFLearn, tqdm, numpy, open-cv, os, random, matplotlib.

b) Loading the data:- Then we loaded the training data. It took 4 5 mins to load the data as the size of the data is huge. Then we split the data into training and testing sets. We used 80% data for training and 20% for testing.

c) Defining the model:- Then we define a convolutional neural network (CNN) architecture using the Keras API in TensorFlow.

Then firstly we created an input layer for the model with a shape of (300, 300, 3), which indicates that each input image is of size 300x300x3, where 3 represents the three color channels - red, green, and blue.

Then we defined the convolutional layers, which extract features from the input images. Each layer uses a 3x3 kernel to convolve over the input image and extract a set of filters, which are then passed through the 'Rectified Linear Unit' (ReLU) activation function which is a common activation function used in neural networks, to introduce non-linearity. Each convolutional layer is followed by a max-pooling layer with a 2x2 window that reduces the spatial size of the output and helps in preventing overfitting.

The layers in this particular CNN are as follows:

1. The first convolutional layer applies 16 filters that are each 3x3 in size. The activation function used is the rectified linear unit (ReLU).

2. The output of the first convolutional layer is then passed through a max-pooling layer with a 2x2 window. This reduces the spatial dimensions of the output by a factor of 2.

3. The second convolutional layer applies 32 filters that are each 3x3 in size. The output is again passed through a max-pooling layer with a 2x2 window.

4. The third convolutional layer applies 64 filters that are each 3x3 in size. The output is again passed through a max-pooling layer with a 2x2 window.

5. The fourth convolutional layer applies 128 filters that are each 3x3 in size. The output is again passed through a max-pooling layer with a 2x2 window.

6. After the convolutional and pooling layers, the output (feature map) is flattened to a 1-dimensional tensor, which can then be passed through a fully connected layer. The fully connected layer has 512 hidden units and uses the ReLU activation function.

7. The fully connected layer is created using the Dense function, which specifies the number of units (512) and the activation function (ReLU). This layer applies a matrix multiplication operation to the flattened tensor followed by the ReLU activation function.

8. Finally, the output layer of the network has 5 nodes, one for each class in the classification task. The activation function used for the output layer is 'Softmax', which normalizes the output so that the sum of the outputs is 1, and each output can be interpreted as the probability of the input image belonging to that particular class.[6]

The entire network is defined using the 'Model' function that takes the input layer and output layer as arguments. This model can then be trained on the training data and evaluated on the testing data to perform the image classification task.

https://doi.org/10.46335/IJIES.2023.8.4.1 **e-ISSN: 2456-3463**
Vol. 8 , No. 4, 2023, PP. 1-7

d) Compiling the model:- After building the model, the next step is to compile it, which means specifying the loss function, optimizer, and metrics to be used during training.

In this case, the loss function is specified as 'sparse_categorical_crossentropy', which is a commonly used loss function for multiclass classification problems. The optimizer used is 'Adaptive Moment Estimation' (Adam), which is a popular stochastic gradient descent optimizer that adjusts the learning rate adaptively. The learning rate is set to 0.00005.[2]

Finally, the metric used to evaluate the performance of the model during training is accuracy. This metric is used to calculate the percentage of correctly classified images in the training set.

e) Data preparation for training and testing: Now after compiling the model the data is prepared for the training and testing of the CNN model. The input images and corresponding labels are separated into training and testing sets.

The training data is represented by two lists, let's say, 'training_X' and 'training_Y', where each element of the list 'training_X' contains an image and each element of the list 'training_Y' contains the corresponding label. Similarly, the testing data is represented by two lists, 'testing_X' and 'testing_Y.'

To prepare the data for training, the images are extracted from the lists and converted to NumPy arrays using the 'np.array()' function. For the training set, the input images are extracted from the training_X list using list comprehension and stored in, let's say, X_train. Similarly, the labels are extracted from the training_Y list and stored in Y_train.

For the testing set, the input images are extracted from the testing_X list using list comprehension and stored in, let's say, 'X_test'. Similarly, the labels are extracted from the testing_Y list and stored in 'Y_test.'

f) Training the data: Now using this prepared data the model is trained on the training dataset X_train and Y_train and the performance of the model is evaluated on the testing dataset X_test and Y_test.

For this 'model.fit()' method of the Sequential class in 'Keras' that train the model for a fixed number of epochs (iterations on a dataset) is used.

Here are the arguments passed to the model.fit():

1. X_train: This is the training input data. It is a NumPy array of shape (2930, 300, 300, 3), where 2930 is the number of training samples, 300 is the width and height of each image, and 3 is the number of color channels (RGB).

2. Y_train: This is the training target data. It is a NumPy array of shape (2930,), where 2930 is the number of training samples.

3. batch_size: This is the number of samples that will be used in each training batch. In this case, 64 samples will be used in each batch. Basically, batch size refers to the number of training samples that are propagated through a neural network in a single forward/backward pass.

4. epochs: This is the number of times the model will be trained on the entire training dataset. In this case, the model will be trained for 10 epochs. One epoch consists of one or more batches, where each batch is a subset of the training dataset.

5. verbose: This determines the amount of information printed during training. A value of 1 means that progress bars will be displayed.

6. Validation data: This is the data on which we evaluate the loss and any model metrics at the end of each epoch. In this case, X_test and Y_test are used for validation.

4) Results &Model Evaluation: After this, the next task is evaluating the performance of the trained model on the testing dataset. The 'model.evaluate' method is used to calculate the loss and accuracy of the model on the testing dataset. The goal of training a machine learning model is to minimize the loss and maximize the accuracy.

The method takes the testing dataset X_test and Y_test as input and returns the values of loss and accuracy on this dataset.

The test accuracy percentage was found to be approximately 93% and the test loss percentage was found to be approximately 18%.

5) Visualization:- Now we plot the training and validation accuracy and loss of the model during the training process.

First, the code extracts the training and validation accuracy, loss, and number of epochs from the 'Model_fit'

https://doi.org/10.46335/IJIES.2023.8.4.1 e-ISSN: 2456-3463
Vol. 8 , No. 4, 2023, PP. 1-7

method's 'history' attribute.

Then, the code creates two subplots for accuracy and loss separately. In the first subplot, the training accuracy and validation accuracy are plotted against the number of epochs. The green line represents the training accuracy, and the blue line represents the validation accuracy.

In the second subplot, the training loss and validation loss are plotted against the number of epochs. Again, the green line represents the training loss, and the blue line represents the validation loss.

Finally, the code displays the two plots with a legend and a title. These plots give us an idea of how well the model is performing on the training and validation data and help us to diagnose potential issues such as overfitting or underfitting.

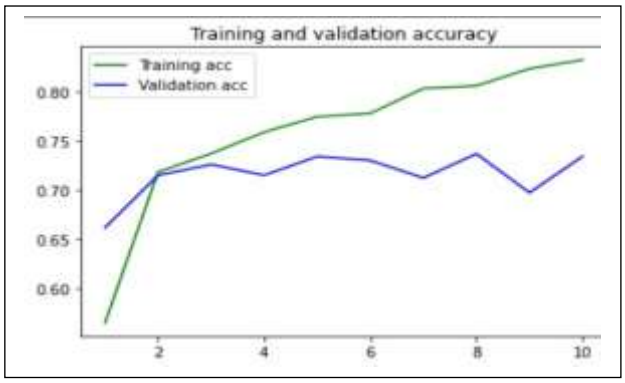

Fig. 1.The training and validation accuracy plotted against the number of epochs.

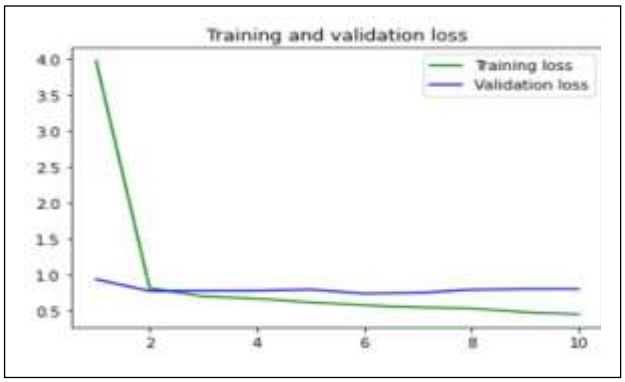

Fig. 2.The training and validation loss plotted against the number of epochs.

So, this is how we implemented CNN for our project.

V. Comparative Study:-

A. Accuracy:

CNN is found to achieve higher accuracy compared to SVM for DR detection. CNN can learn hierarchical representations of the input data and extract complex features, leading to high accuracy in detecting DR from retinal fundus images. SVM, on the other hand, relies on handcrafted features and does not perform as well as CNN.

B. Dataset size:

CNN requires large datasets for training and optimization, while SVM can perform well with smaller datasets. With the availability of large public datasets like Messidor and Kaggle, CNN can be trained effectively for DR detection. However, SVM can be a good choice when only a limited amount of labeled data is available.

C. Preprocessing:

Preprocessing techniques such as image normalization and feature extraction are important for effective DR detection. CNN can learn feature representations from the raw input data, while SVM relies on handcrafted features extracted from the preprocessed images. Therefore, CNN can be more effective in detecting DR from retinal fundus images with complex features.

D. Computational complexity:

CNN is computationally more complex than SVM and they require significant computational resources for training and optimization. SVM, on the other hand, is computationally efficient and can be trained on relatively modest hardware.

E. Interpretability:

SVM is more interpretable than CNN since they rely on handcrafted features that can be easily understood. CNN, on the other hand, learns complex feature representations that may be difficult to interpret.

In summary, CNN is found to achieve higher accuracy compared to SVM for DR detection. However, SVM can be a good choice when only a limited amount of labeled data is available, or when a computationally efficient algorithm is required. Ultimately, the choice of algorithm for DR detection depends on the specific requirements of the application and the available resources.

VI. Conclusions:

The results showed that the CNN model outperformed the SVM model. Therefore, it can be concluded that using CNN for diabetic retinopathy detection is a promising approach. However, more research is required to further optimize the CNN model and to explore other deep learning techniques for this purpose. Overall, this study provides valuable insights for improving the accuracy and efficiency of diabetic retinopathy detection using deep learning methods.

https://doi.org/10.46335/IJIES.2023.8.4.1 **e-ISSN: 2456-3463**
Vol. 8 , No. 4, 2023, PP. 1-7

VII. FUTURE WORK

After carrying out this project work, the following recommendations were made by our guide and other faculties, based on the limitations encountered during the implementation:-

1. Research can be extended to further optimize the CNN model and explore other deep learning techniques for this purpose.

2. More precise and customized reports can be made as per the doctor's need.

3. This model then can be commercialized.

ACKNOWLEDGMENT

We hereby express our sincere gratitude to our professor from the Department of Information Technology, Dr. Roopali Lolage for guiding us throughout this project work. We are heartily thankful for her valuable guidance.

REFERENCES

[1] *Mateen, Muhammad; Wen, Junhao; Hassan, Mehdi; Nasrullah, Nasrullah; Sun, Song; Hayat, Shaukat (2020). Automatic Detection of Diabetic Retinopathy: A Review on Datasets, Methods and Evaluation Metrics. IEEE Access, Volume 8, pages: 48784 - 48811, Issue 11, March 2020.*

[2] *Zhixiang Qian, Chenjian Wu, Hong Chen, Minxin Chen, "Diabetic Retinopathy Grading Using Attention based Convolution Neural Network",2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) |978-1-72818028-1/20/$31.00 ©2021 IEEE | DOI: 10.1109/IAEAC508 56.2021.9390963.*

[3] *Shital N. Firke, Ranjan Bala Jain, "Convolutional Neural Network for Diabetic Retinopathy Detection",2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) | 978-1-7281-9537-7/20/$31.00 ©2021 IEEE |DOI:10.1109/ICAIS50930.2021.9395796.*

[4] *Prof. Ketki C. Pathak, Riddhi B. Shah, Reshma R. Tharakan, Bhavya N. Patel, Dhruvi C. Jariwala," **Diabetic Retinopathy Diagnosis and Categorization using Deep Learning - A Review**", Proceedings of the Fifth International Conference on Intelligent Computing and Control Systems (ICICCS 2021) IEEE Xplore Part Number: CFP21K74-ART; ISBN: 978-0-7381-1327-2.*

[5] *K. Shankar1, Yizhuo Zhang2, Yiwei Liu2, Ling Wu2, and Chi-Hua Chen2, Senior Member, IEEE, "Hyperparameter Tuning Deep Learning for Diabetic Retinopathy Fundus Image Classification ", DOI 10.1109/ACCESS.2020.3005152, IEEE Access.*

[6] *Nagaraj G, Sumanth Simha C, Harish Chandra G R, Dr Indiramma M, Professor," Deep Learning Framework for Diabetic Retinopathy Diagnosis", Third International Conference on Computing Methodologies and Communication (ICCMC 2019) IEEE Xplore Part Number: CFP19K25-ART; ISBN: 978-1-5386-7808-4.*

[7] *Mahendran Gandhi, Dr. R. Dhanasekaran," Diagnosis of Diabetic Retinopathy Using Morphological Process and SVM Classifier", International conference on Communication and Signal Processing, April 3-5, 2013, India, 978-1-4673-4866-9/13/$31.00 ©2013 IEEE.*