2016 International Conference on Computing, Analytics and Security Trends (CAST)
College of Engineering Pune, India. Dec 19-21, 2016

Diabetic Retinopathy Detection using Deep Convolutional Neural Networks

Darshit Doshi
Sardar Patel Institute of Technology
darshit.doshi@gmail.com

Aniket Shenoy
Sardar Patel Institute of Technology
shenoyaniket95@gmail.com

Deep Sidhpura
Sardar Patel Institute of Technology
deepsidhpura777@gmail.com

Dr. Prachi Gharpure
Sardar Patel Institute of Technology
prachigharpure@spit.ac.in

***Abstract*— Diabetic retinopathy is when damage occurs to the retina due to diabetes, which affects up to 80 percent of all patients who have had diabetes for 10 years or more. The expertise and equipment required are often lacking in areas where diabetic retinopathy detection is most needed. Most of the work in the field of diabetic retinopathy has been based on disease detection or manual extraction of features, but this paper aims at automatic diagnosis of the disease into its different stages using deep learning. This paper presents the design and implementation of GPU accelerated deep convolutional neural networks to automatically diagnose and thereby classify high-resolution retinal images into 5 stages of the disease based on severity. The single model accuracy of the convolutional neural networks presented in this paper is 0.386 on a quadratic weighted kappa metric and ensembling of three such similar models resulted in a score of 0.3996.**

***Index Terms*— Diabetic Retinopathy, Computer vision, Deep learning, Convolutional Neural Networks, Quadratic weighted kappa metric,**

I. INTRODUCTION

A. Diabetic Retinopathy

Diabetic retinopathy (DR), also known as diabetic eye disease, is when damage occurs to the retina due to diabetes. It can eventually lead to blindness. It is an ocular manifestation of diabetes. Despite these intimidating statistics, research indicates that at least 90% of these new cases could be reduced if there were proper and vigilant treatment and monitoring of the eyes. The longer a person has diabetes, the higher his or her chances of developing diabetic retinopathy. Diabetic retinopathy can be diagnosed into 5 stages: mild, moderate, severe, proliferative or no disease. The various signs and markers of diabetic retinopathy include microaneurysms, leaking blood vessels, retinal swellings, growth of abnormal new blood vessels and damaged nerve tissues [7].

DR detection is challenging because by the time human readers submit their reviews, often a day or two later, the delayed results lead to lost follow up, miscommunication, and delayed treatment. Clinicians can identify DR by the presence of lesions associated with the vascular abnormalities caused by the disease. While this approach is effective, its resource demands are high. The expertise and equipment required are often lacking in areas where the rate of diabetes in local populations is high and DR detection is most needed. The need for a comprehensive and automated method of DR screening has long been recognized, and previous efforts have made good progress using image classification, pattern recognition, and machine learning [7].

The current research in diagnosing diabetic retinopathy has been based on explicit extraction of features like microaneurysms and lesions through which the classification is performed. There has also been research in using machine learning techniques to classify the image as normal or diseased [14, 15, 16].

This paper aims at proposing a diabetic retinopathy diagnosis model that automatically learns features which are pivotal in diagnosing the stage of the disease without explicit or manual feature extraction.

Fig. 1: Normal Vision[7]

B. Convolutional Neural Networks

Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition (Krizhevsky et al., 2012 [1]; Zeiler & Fergus, 2013; Sermanet et al., 2014; Simonyan & Zisserman, 2014) which has become possible due to the large public image repositories, such as ImageNet (Deng et al., 2009), and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al., 2012). In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition

978-1-5090-1338-8/16/$31.00 ©2016 IEEE

Fig. 2: Vision with diabetic retinopathy[7]

Challenge (ILSVRC) (Russakovsky et al., 2014), which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature encodings (Perronnin et al., 2010) (the winner of ILSVRC-2011) to deep ConvNets (Krizhevsky et al., 2012) (the winner of ILSVRC-2012) [3]. The performance of convolutional neural networks in these competitions was the motivation behind adopting CNN for this research.

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D hierarchical structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units [6]. CNNs also consider the hierarchical representation of images while training by stacking multiple trainable stages on each other [2].

II. The Dataset

The dataset consists of 35,126 labeled high-resolution colour fundus retinal images belonging to five classes corresponding to the five stages of the disease as portrayed in Table I. The test set consists of 53,576 images out of which 5,000 have been utilized for this paper. The images have been open-sourced by EyePACs, a free platform for retinopathy screening. A trained clinician has rated the presence of diabetic retinopathy in each image on a scale of 0 to 4 [5].

The images in the dataset come from different models and types of cameras, which can affect the visual appearance of left and right retinas. Some images are shown as one would see the retina anatomically (macula on the left, optic nerve on the right for the right eye). Others are shown as one would see through a microscope condensing lens (i.e. inverted, as one sees in a typical live eye exam). There is also noise in both the images and labels. Images may contain artifacts, be out of focus, underexposed, or overexposed and are of different resolutions. Table 1 shows the distribution of image classes in the dataset.

TABLE I: Class Distribution in Original Dataset

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%

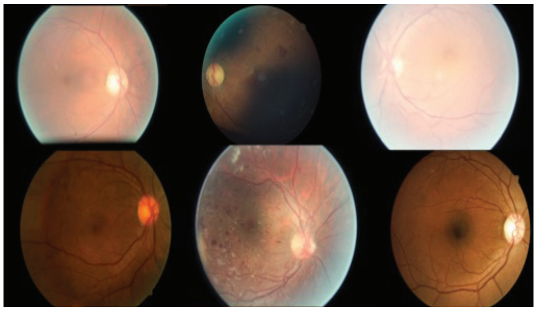

Fig. 3: Sample Dataset Images

III. Data Pre-processing

Due to non-standard image resolutions, the training images could not be utilized directly for training. The images were scaled down to a fixed resolution size of 512x512 pixels to form a standardized dataset.

Training images of resolution 512x512 pixels on all three colour channels demanded high memory requirements. Due to this limitation, the images were converted to a single channel. After several experiments, it was found that green channel images retained information better than the other channel images.

In order to enhance the contrast of the image evenly across pixels, histogram equalization technique was applied on the images.

In order to prevent the convolutional neural network from learning the inherent background noise in the image, each image was normalized using Min-Max normalization.

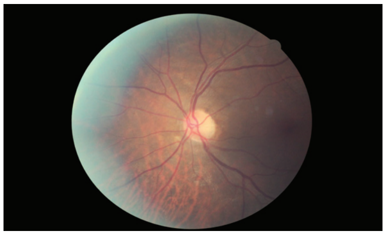

Fig. 4: Original Image

IV. The CNN Architecture

The multiple architectures of the deep network is shown in Table II. The network contains an input layer which takes images with resolution 512x512 pixels as input. The architecture consists of 5 sets of combination of convolution,

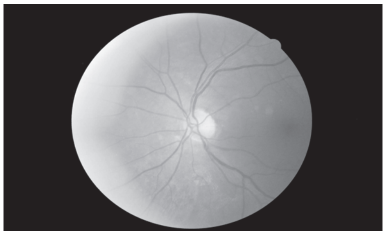

Fig. 5: Green Channel Image

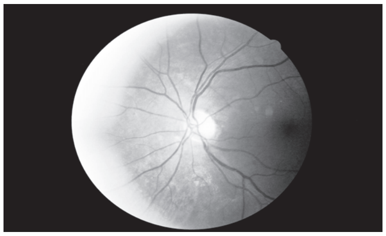

Fig. 6: Green Channel Image with Histogram Equalization

pooling and dropout layers, each stacked one over the other. This is followed by 2 sets of fully connected hidden and feature pooling layers. This is followed by the final output layer.

To preserve the spatial size of the input and output volumes, zero-padding was utilized in the convolutional layers.

A. Convolutional Layer

In the architectures, one convolutional layer is stacked over another without spatial pooling between them. According to Andrew Zisserman et al. [3], this allows for a larger receptive field size, makes the decision function more discriminative and also decreases the number of learnable parameters, thus allowing architectures with increased depth.

The first convolutional layer consists of 16 filters, each of the size 5x5. Each filter convolves over the input image with stride as 2.

The third convolutional layer consists of 32 filters, each of size 3x3. Each filter convolves over the output of the previous layer with stride as 2.

All the remaining convolutional layers had filters and convolved matrix size as specified in Table II. Each filter convolves with stride as 1.

B. Pooling Layer

Each pooling layer utilized max-pooling with filter of size 2x2. The max-pooling layer will perform a MAX operation which takes the maximum over the 2x2 region in the depth slice of the input.

C. Dropout Layer

The role of dropouts in preventing a large network from overfitting and providing performance improvements of neural networks in supervised learning tasks is proposed in [10].

The dropout ratio in each dropout layer from the first to the fourth layer was increased uniformly from 0.1 to 0.4. The dropout ratio in the fifth and the sixth layer were kept fixed at 0.5.

TABLE II: CNN Architectures of 3 models

Layers	Model 1	Model 2	Model 3
input	1x512x512	1x512x512	1x512x512
conv 1	16x256x256	16x256x256	16x256x256
conv 2	16x256x256	16x256x256	16x256x256
pool 1	16x128x128	16x128x128	16x128x128
dropout 1	16x128x128	16x128x128	16x128x128
conv 3	32x64x64	32x64x64	32x64x64
conv 4	32x64x64	32x64x64	32x64x64
pool 2	32x32x32	32x32x32	32x32x32
dropout 2	32x32x32	32x32x32	32x32x32
conv 5	48x32x32	64x32x32	64x32x32
conv 6	48x32x32	64x32x32	64x32x32
conv 7	48x32x32	64x32x32	64x32x32
pool 3	48x16x16	64x16x16	64x16x16
dropout 3	48x16x16	64x16x16	64x16x16
conv 8	64x16x16	128x16x16	96x16x16
conv 9	64x16x16	128x16x16	96x16x16
conv 10	64x16x16	128x16x16	96x16x16
pool 4	64x8x8	128x8x8	96x8x8
dropout 4	64x8x8	128x8x8	96x8x8
conv 11	128x8x8	256x8x8	128x8x8
conv 12	128x8x8	256x8x8	128x8x8
pool 5	128x4x4	256x4x4	128x4x4
dropout 5	128x4x4	256x4x4	128x4x4
hidden 1	400	256	256
maxout 1	200	128	128
dropout 6	200	128	128
hidden 2	400	256	256
maxout 2	200	128	128
output	5	5	5

input - Input layer (no. of images x dimensions of image); conv - Convolution layer (no. of filters x dimensions of convolved matrix); pool - Pooling layer (dimension of pooled matrix); dropout - Dropout layer; hidden - Fully connected hidden layer (no. of hidden units); maxout - Feature Pooling layer; output - Output layer (output units corresponding to the 5 classes).

D. Hidden Layers and Feature Pooling Layers

The number of fully connected units in both the hidden layers was fixed at 400 for the first model and decreased to 256 for the second and third model.

Each hidden layer was followed by a maxout implemented through a feature pooling layer with filter size of 2, which pools the features of the hidden units, incorporating the beneficial characteristics of optimization and model averaging with dropouts [9].

E. Activation Functions

The activation functions used for the convolutional layers and hidden layers was Leaky Rectifier. Compared to the standard rectifier, it has a non-zero gradient for negative input which often helps in convergence [11].

Leaky rectifier $\varphi(x) = max(\alpha.x, x)$ where α is the leakiness parameter and x is the input. A leakiness of 0 will lead to a standard rectifier and a leakiness of 1 will lead a linear activation function. In above models, the leakiness parameter was set to 0.01.

The output layer utilizes a 5 way softmax activation function which produces a probability distribution over the 5 classes.

TABLE III: Class Distribution after Class Balancing

Class	Name	No. of images	Percentage
0	No DR	25810	35.65%
1	Mild DR	12215	16.87%
2	Moderate DR	26460	36.55%
3	Severe DR	4365	6.02%
4	Proliferative DR	3540	4.89%

V. Details of Learning

Having established the background of the paper and the architecture of the convolutional neural network, this section will deal with the various details and parameters of the learning process.

The deep network described above was designed keeping in mind the complexity of the features involved in diagnosing the disease. Such a deep network would have over-fitted with the original dataset. To prevent this, the dataset was augmented using image transformations such as shear, flop, transverse and transpose.

Table I depicts the high class imbalance in the original dataset. Such imbalance would result in over-fitting of the majority classes (class 0 and class 2) and under-fitting of the other classes. To counter this, the classes had to be roughly balanced. This was achieved by dropping the transformed images belonging to class 0. This resulted in the final training dataset consisting of 72390 images with the distribution as shown in Table III. Though this distribution is still unbalanced, it is better balanced than the original dataset. This method resulted in a higher quadratic kappa score as it prevented over-fitting of class 0 and class 2 data.

To initiate the training process, it was essential to find the ideal batch size and mini-batch size that would fit the RAM and GPU, respectively. The batch size denotes the number of images that can be read into the RAM for the training process. The mini-batch size denotes the number of images that are transferred from the RAM to the GPU for computations.After several experimentations with the parameters of the network (such as stride, filter size, etc.), it was found that a mini-batch size of 64 was ideal for the network described above. This resulted in a batch size 2413 images.

For this multi-classification problem which has softmax output units, a categorical Cross-Entropy loss function was chosen. The function below denotes the categorical cross-entropy between the classes and targets:

$$\L_i = -\sum\nolimits_j t_{i,j}\, log(p_{i,j}) \tag{1}$$

To minimize the loss function mentioned above, the models were trained using Nesterov Momentum along with Stochastic Gradient Descent with a mini-batch size as mentioned previously. The importance of using the above technique as a gradient update method was shown in [12]. The number of epochs for the training process was 250. This update rule is described as follows:

$$\begin{aligned} velocity := (momentum * velocity) - \\ (learning\_rate * gradient) \end{aligned} \tag{2}$$

$$weight := weight + velocity \tag{3}$$

In the above update rule, the momentum parameter was set to 0.9 throughout the training. In order to remove the fluctuations in final weights caused due to variations in batches, an adaptive learning rate relative to the number of epochs was employed instead of a fixed learning rate. The adaptive learning rate schedule was 0.003 for the first 200 epochs and 0.0003 for the remaining 50 epochs.

The weights and biases were initialized using Glorot-style initialization, sampled from the uniform distribution. This helps in faster convergence of the objective function [13].

Over-fitting is a common problem faced in deep networks. To control over-fitting, in addition to data augmentation, dropouts were utilized in each layer with the ratios described in the network previously.

VI. Implementation Details

The various data pre-processing techniques were performed using ImageMagick (a command line tool for image processing) and the Python library OpenCV.

To speed up the process of training the above convolutional neural network, the entire training process was performed using NVIDIA GeForce GTX 780 series GPU of RAM 3GB hosted in a machine with Intel core i3 processor and 8GB RAM. In order to interface with the GPU, Theano library [17, 18] was used. To design and train the deep network above, the Python libraries Lasagne [19] and Nolearn [4] were utilized.

VII. Results

In this research, it is not only important to measure the number of correctly and incorrectly classified test images, but also to evaluate by how many classes the images were misclassified and to penalize the accuracy score accordingly.

Thus results were evaluated using a quadratic weighted kappa metric, which measures the agreement between two ratings. This metric typically varies from 0 (random agreement between raters) to 1 (complete agreement between raters). In the event that there is less agreement between the raters than expected by chance, this metric may go below 0. The quadratic weighted kappa is calculated between the scores assigned by the human rater and the predicted scores [8].

Images have five possible ratings, 0,1,2,3,4. Each image is characterized by a tuple(ea,eb), which corresponds to its scores by Rater A (human) and Rater B (predicted). The quadratic weighted kappa is calculated as follows. First, an N x N histogram matrix O is constructed, such that Oi,j corresponds to the number of images that received a rating I by A and a rating j by B. An N-by-N matrix of weights, w, is calculated based on the difference between raters' scores:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{4}$$

An N-by-N histogram matrix of expected ratings, E, is calculated, assuming that there is no correlation between rating scores. This is calculated as the outer product between each rater's histogram vector of ratings, normalized such that E and O have the same sum. From these three matrices, the quadratic weighted kappa is calculated as:

$$k = 1 - \frac{\sum_{i,j} w_{i,j} *O_{i,j}}{\sum_{i,j} w_{i,j} *E_{i,j}} \tag{5}$$

After training Model 1 as portrayed in Table II, on the dataset, and testing it on the 5,000 images in out test set, a quadratic kappa score of 0.3066 was obtained. The large number of hidden units might have resulted in slight overfitting of the model. To further validate this assumption, the hidden units were decreased to 256 in Model 2. The number of filters in the latter convolutional layers were also increased.

On training and testing Model 2 as explained above, a quadratic kappa score of 0.35 was obtained which was a considerable improvement over Model 1. Decreasing the number of hidden units helped increasing the score, however increase in the number of filters might have increased the complexity of the network, thus hampering generalization of the model on the test set. Thus the number of filters in the latter convolutional layers were decreased in Model 3.

On training and testing Model 3, a quadratic kappa score of 0.38659 was obtained. Again, this score was an improvement over the previous model.

In order to incorporate the best learnt features of the above three models and to average out the errors, ensemble averaging was performed. After experimenting with different weights, the best quadratic kappa score of 0.3996 was obtained by combining the Models 1, 2, 3 in the weighted ratio 0.075, 0.125, 0.85 respectively.

TABLE IV: Summary of Results

Model	**Quadratic kappa score**
Model 1	0.3066
Model 2	0.35
Model 3	0.386
Ensemble	0.3996

The Fig. 7 shows the comparison between the number of actual labels of the test images and the predicted labels from the ensembled model for each class. It also shows the number of images that were misclassified by 1 class for each of the 5 classes.

VIII. CONCLUSION

This paper presents the design, architecture and implementation of deep convolutional neural networks for automatic detection and classification of diabetic retinopathy from color fundus retinal images. It also discusses the qudratic kappa metric used to evaluate the prediction results. This research involves three major CNN models, designing their architectures and finding the corresponding quadratic kappa scores. The best score of 0.3996 is obtained by the ensemble of these three models.

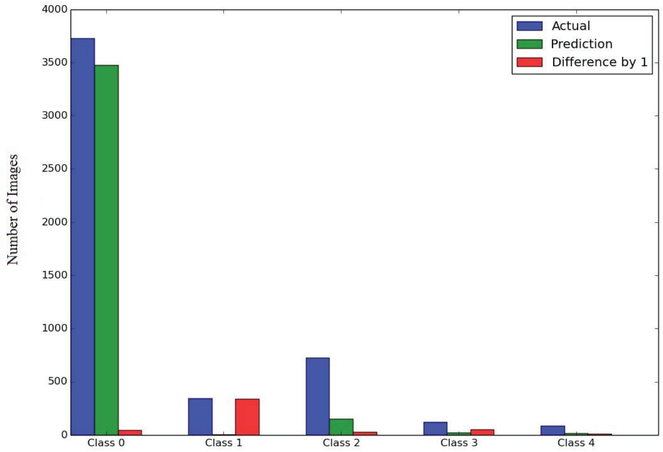

Fig. 7: Visualization of Predictions

IX. FUTURE SCOPE

- Fine tuning the current network parameters to obtain a greater accuracy on single channel images.
- Using all the channels instead of a single channel enabling the network to learn more features thereby decreasing over-fitting through increasing complexity of data.
- Working with alternate image pre-processing techniques to improve noise reduction

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Network. Advances in Neural Information Processing System 25, 2012

[2] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253256. IEEE, 2010.

[3] Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015.

[4] D. Nouri(2014, Dec 17). *Using Convolutional Neural Nets To Detect Facial Keypoints Tutorial* [Online]. Available : http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/

[5] *Diabetic retinopathy image dataset* [Online]. Available: https://www.kaggle.com/c/diabetic-retinopathy-detection/data

[6] *Unsupervised Feature Learning and Deep Learning Tutorial by Stanford University* [Online]. Available: http://ufldl.stanford.edu/tutorial/

[7] *Diabetic retinopathy* [Online]. Available : https://en.wikipedia.org/wiki/Diabetic_retinopathy

[8] *Quadratic Weighted Kappa Metric* [Online]. Available : https://www.kaggle.com/c/diabetic-retinopathy-detection/details/evaluation

[9] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, Yoshua Bengio. Maxout Networks arXiv:1302.4389v4 [stat.ML] 20 Sep 2013

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014).

[11] Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.

[12] Ilya Sutskever1, James Martens, George Dahl, Geofrey Hinton. On the importance of initialization and momentum in deep learning. Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.

[13] Xavier Glorot, Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010. Volume 9 of JMLR.

[14]] R.Priya, P. Aruna. Diagnosis of Diabetic Retinopathy using Machine Learning Techniques. ICTACT Journal on Soft Computing 2013.

[15] Gilbert Lim, Mong Li Lee, Wynne Hsu, Tien Yin Wong. Transformed Representations for Convolutional Neural Networks in Diabetic Retinopathy Screening. Modern Artificial Intelligence for Health Analytics: Papers from the AAAI-14.

[16] Sohini Roychowdhury, Dara D. Koozekanani, Keshab K. Parhi. DREAM: Diabetic Retinopathy Analysis Using Machine Learning. IEEE Journal of Biomedical and Health Informatics, Vol. 18, No. 5, September 2014.

[17] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio. Theano: new features and speed improvements. NIPS 2012 deep learning workshop.

[18] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. Theano: A CPU and GPU Math Expression Compiler. Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. June 30 - July 3, Austin, TX

[19] Zenodo (2015, Aug 13) *Lasagne* [Online]. Available: https://zenodo.org/record/27878#.Vc0Iz3UVhHx