2017 2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT), May 19-20, 2017, India

Detecting Diabetic Retinopathy using Deep Learning

Yashal Shakti Kanungo,
1mv13cs054@sirmvit.edu

Bhargav Srinivasan,
1mv13cs029@sirmvit.edu

Dr. Savita Choudhary,
savitha cs@sirmvit.edu

Abstract— Diabetic Retinopathy (DR) is an eye disease caused mainly due to enduring diabetes and is found to be the principal cause of blindness among the working age population in developed nations. One out of two people suffering from diabetes has been diagnosed with some stage of DR. Detection of DR symptoms in time can avert the vision impairment in majority of cases, however such revelation is difficult with present tools and methods. There has been a need for comprehensive and automated DR detection tools and methods. Previous methods employing image classification, pattern recognition and machine leaning has provided promising results. In the present study, color fundus photography has been used and the results suggest that the model has realistic clinical potential.

I. Introduction

Diabetic Retinopathy (DR) is a medical condition of the eye that is caused by diabetes. It is one of the leading causes for blindness in the working age population. To detect this condition, highly trained medical professionals are needed to study the fundus of the eye, but manual diagnosis is time consuming and costly. Fig. 1 shows the visualization of the medical condition. The diagnosis can be automated by using Convolutional Neural Networks [ConvNets]. Kaggle [1], a platform for predictive modeling and analytics competitions, hosted a Diabetic Retinopathy Detection competition, sponsored by California Health Care Foundation [CHCF]; provided around 80,000 scans of eye fundus to train and test the ConvNet model. We followed one of the high ranked submissions, in which Deep ConvNets was used. Convolution Neural Networks have abilities to identify patterns directly from image at pixel level and hence requires minimal preprocessing. It provides advantage over others by recognizing patterns under extreme variability such as in case of handwritten characters. They are robust and less effected by distortions and geometrical transformations. Fig.2 shows an example of how the data set will be provided.

Fig.1. A comparison of a normal retina and diabetic retinopathy

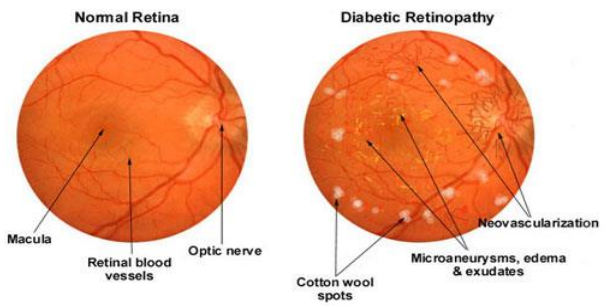
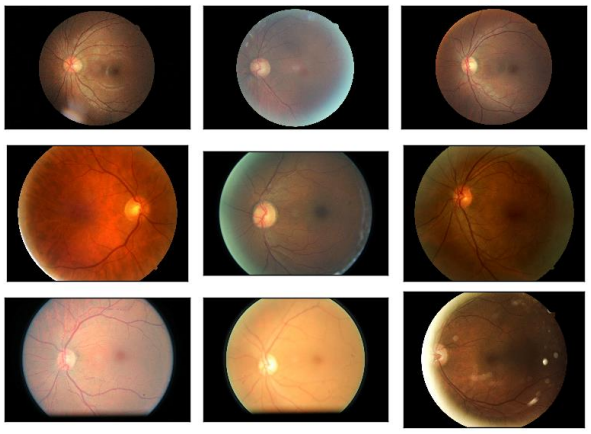


Fig. 2. An example of a random data set. Note the differences in image color, brightness, contrast, etc. in some of the random training images.



II. Literature Survey

There have been a few methods that have been tried in the past and recently in parallel with our workings that aimed to detect Diabetic Retinopathy. The first journal that was surveyed was Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs [2]. In this journal, a deep learning algorithm was used for automated detection of diabetic retinopathy in retinal fundus images. The data set was first graded by a panel of 54 ophthalmologists and each image was graded 3 to 7 images to avoid bias. The deep convolutional neural network had high sensitivity and specificity.

In Rethinking the Inception Architecture for Computer Vision [5], a new method of scaling up networks is introduced. By benchmarking on the ILSVRC 2012 validation set, they achieved 5.6 top-5 errors and with ensemble, achieved a 3.5 top-5 error. Thus, deep learning proved to be an exciting potential which was not fully explored and it motivated us to work on this problem statement using this approach.

III. Architecture Of ConvNet Model

Architecture is the most important aspect of a neural network. It provides the overview of how the model is set up so that it can be replicated with the same specifications. Architecture of the proposed model has the following features.

978-1-5090-3704-9/17/$31.00 © 2017 IEEE

A. Description

The proposed model is built around the Inception-v3 architecture. The architecture basically acts as multiple convolution filter inputs that are processed on the same input. It also does pooling at the same time. All the results are then concatenated. This allows the model to take advantage of multi-level feature extraction from each input. For instance, it extracts general (5x5) and local (1x1) features at the same time.

B. Specifications

The various layers and convolutions that were used are as shown in Table I

TABLE I

INCEPTION-V3

TYPE	PATC	PATCH-2	PATCH-3	SIZE	DEPTH
"conv"	3	3	2	[299,299]	3
"conv"	3	3	1	[149,149]	32
"conv padded"	3	3	1	[147,147]	32
"pool"	3	3	2	[147,147]	64
"conv"	3	3	1	[73,73]	64
"conv"	3	3	2	[71,71]	80
"conv"	3	3	1	[35,35]	192
"3 Inception"				[35,35]	288
"5 Inception"				[17,17]	768
"2 Inception"				[8,8]	1280
"pool"	8	8		[8,8]	2048
"linear"	"logits"			[1,1]	2048
"softmax"	"classifie			[1,1]	1000

C. Design principle behind the Inception-v3 architecture

Figure 3 shows the Inception-v3 architecture. The architecture focuses on not only improving the accuracy and statistical performance of the model but also on the most efficient way to do so considering the training time and memory footprint. Some principles behind the architecture are:

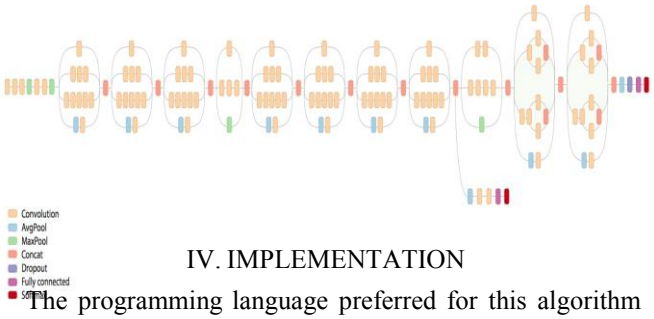
a) Avoiding extreme compression of information: The size of representation should gradually reduce from input side towards the output such that such that by examining the architecture at any cross section between the input and the output layer there should not be any bottleneck with extreme compression.

b) Higher dimensional representations: Local processing has been found to be effective for higher dimensional representation. By increasing the number of activations in a tile, the convolutional network develops more unscramble features and network takes fewer training time.

c) Spatial combination: Spatial combinations can be achieved over lower dimensional vectors without loss in representational power. Given the limitations that these signals should be easily compressible, the dimension reduction even promotes faster learning.

d) Width and Depth optimization for Network balance: High performance network requires an optimization between the number of filters per stage and the depth of the network. High quality networks can be achieved by increasing both width and depth but, optimal improvement can only be achieved if both width and depth are increased in parallel. Hence, it is required to distribute the computational budget between the width and depth. These philosophies as a whole are used to improve the quality of the network. The indication is to use the resources judiciously to enhance the overall network performance.

Fig. 3 - Schematic diagram of Inception-v3



IV. IMPLEMENTATION

The programming language preferred for this algorithm was Python. A number of libraries were used for image processing, machine learning and to build neural networks. The library OpenCV[8] for used for image pre-processing, image loading and image manipulations such as resize and rotation. NumPy[9] was used for mathematical functions needed for machine learning. Theano[10] was used to handle multi-dimensional arrays efficiently and Lasagne[11] was used to define the neural network. A simple menu-driven script was written which gave the users option to train and evaluate the network. Different scripts were written to train the neural network, test the model for an image or evaluate the model on multiple images.

To train the network, a Python function was written which returns the network description and uses that to pass images through. A weights file was created which will hold all the parameters that is learned by the neural network. After training for a number of epochs, the weights file (or the model) can be used further to test future images.

To make loading of the images easier, a loader function was written which loads images while another set of images are passing through the network in parallel. This makes use of a CPU's multi-threading capability. A single-thread image loader was also available if parallel loading was not required.

V. EXPERIMENTS AND RESULTS

The ConvNet is trained on different data sets throughout the experiment. However, the testing was done on all the Kaggle images that were provided as the test data set.

Following are the results of changing each hyperparameter individually and in isolation.

A. Batch Size

To minimize the gradient matrix, so that less memory is utilized in the learning process, the training set is divided into batches, which are then used to train the network. This division into batches is done for a number of iterations until we are satisfied with the validation accuracy and validation loss. Different batch sizes were compared to see how they affect the final weights. Two batch sizes were used; 64 and 128, keeping the other parameters same. Results are shown in Table II.

TABLE II

CHANGES IN BATCH SIZE

Batch size	Sensitivity	Specificity	Accuracy	Unclassified
64	94	81	82	20
128	97	87	88	25

B. Epoch

An epoch is the number of iterations that would ideally cover the complete training set. If you have 1000 training elements, and your batch size is 500, then it will take 2 iterations to complete 1 epoch. Usually, a number of epochs are needed for training to be more effective. Results are shown in Table III.

TABLE III

EPOCH

Epoch	Sensitivity	Specificity	Accuracy	Unclassified
100	72	69	70	30
200	97	87	88	25

C. Preprocessing

Preprocessing is the stage before training. It consists of altering the image that is to be the example for training so that a standard is followed throughout the image set and some features can be highlighted even before training. Some of the common preprocessing techniques are normalization of the data, mean subtraction and Principle Component Analysis (PCA). Results are shown in Table IV.

TABLE IV

PREPROCESSING

	Sensitivity	Specificity	Accuracy	Unclassified
No Preprocessing	93	85	82	17
Preprocessing	97	87	88	25

D. Train dataset size

The train dataset contains all the images that are labeled and is used during the training phase of the experiment. Results are shown in Table V.

From the experiments, the inferences are as follows:

TABLE V

TRAIN DATA SET SIZE

Dataset size	Sensitivity	Specificity	Accuracy	Unclassified
10,000	75	65	67	28
40,000	97	87	88	25

- Larger batch sizes perform better than smaller batch sizes but they need more computing resources in order to run.
- A model that runs for more epochs have a better performance since they train on the same images multiple times and the probability of it making a mistake lowers every epoch. However, there will be a point where further epochs will not change the accuracy and the performance becomes stagnant.
- Preprocessing is generally a good idea as the model will have a better understanding of the input data and will lead to less bias in the network
- Larger train datasets improve the performance of the model since it can learn many variations and edge cases of the problem, making it versatile.

Fig. 4. An example ROC curve for testing

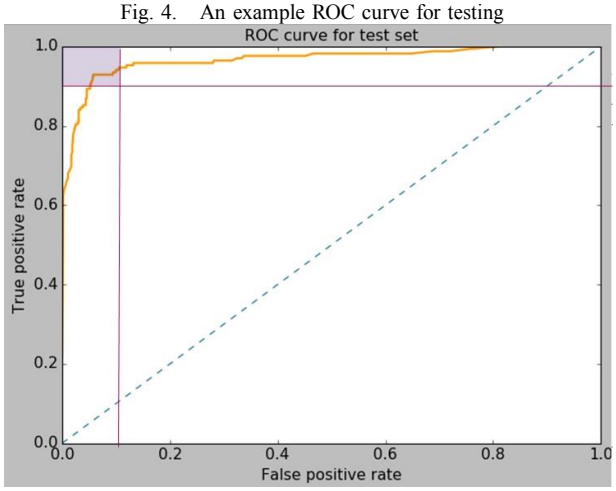
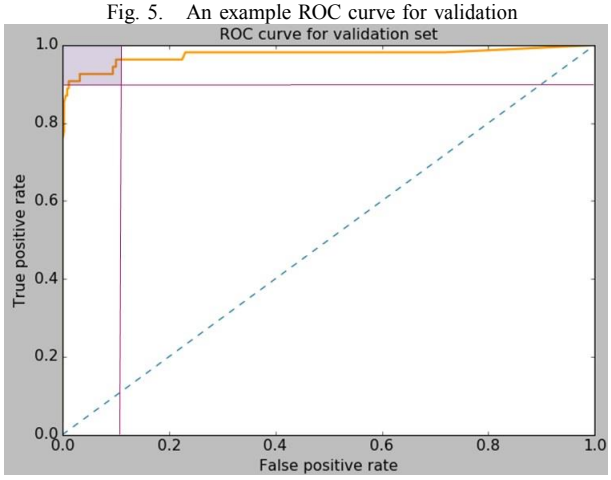


Fig. 5. An example ROC curve for validation



VI. Future Work

With the advent of new neural networks, such as Residual Neural Network by Facebook [7] and better pooling methods, the possibility of tweaking hyper parameters is constantly increasing. The observations presented in this paper are currently being extended to these broader set of methods. Feasible scope of future modifications includes:

- Training multiple architectures and combining them in an ensemble.
- Utilizing multiple images per patient.
- Test-time augmentation, which is the technique of predicting for multiple transformed versions of a test image and averaging the predictions. Using this method, the algorithm will have a better understanding of the input image since it views the image in many transformed views like rotation, scaling and shearing.

VII. Conclusions

It can be thus be inferred from the results of our experiments that the hyper-parameters played a key role in the performance of a model and deriving concrete methods to find the optimal hyper-parameters quickly is an important task at hand. The trained model achieved respectable scores considering the limited nature of training data that is avail- able. The effect of hyper-parameters as well as the quality and quantity of training data is also evident from the fact that Inception-v3 achieves considerably higher scores on the ImageNet challenge owing to the immense training data-set that is available. Discrepancies in the human judgment for corner medical cases are another issue that alters the factual data and thus gives sub-optimal results.

References

[1] "Kaggle: Your Home for Data Science", Kaggle.com, 2017. [Online]. Available: https://www.kaggle.com/.
[2] Gulshan, V., Peng, L., Coram, M., Stumpe, M., Wu, D., & Narayanaswamy, A. et al. (2016). Development and Valida- tion of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. JAMA, 316(22), 2402. http://dx.doi.org/10.1001/jama.2016.17216
[3] "tensorflow/models", Google, 2017. [Online]. Available: https://github.com/tensorflow/models/tree/master/inception.
[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014
[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Re- thinking the inception architecture for computer vision". arXiv preprint arXiv:1512.00567, 2015.
[6] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception- resnet and the impact of residual connections on learning". arXiv:1602.07261, 2016.
[7] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[8] OpenCV Library opencv.org.
[9] NumPy, http://www.numpy.org/.
[10] Theano, http://deeplearning.net/software/theano. [11]
Lasagne, http://lasagne.readthedocs.io/en/latest.