# Detection of Diabetic Retinopathy

Shikha Agarwal
UMass, Amherst
Department of Computer Science
shikhaagarwa@cs.umass.edu

Sneha Bhattacharya
UMass, Amherst
Department of Computer Science
snehabhattac@cs.umass.edu

## Abstract

*Convolutional neural network have been used to detect the presence of diabetic retinopathy from a dataset of Fluorescein Angiography photographs. The dataset was obtained from kaggle where it was provided by EyePacs. A VGG19 network trained on this image achieved an accuracy of 74% and sensitivity of 77%.*

## 1. Introduction

Diabetic retinopathy(DR) is caused by damage to the blood vessels in the tissue at the back of the eye (or retina). It is associated with long-standing diabetes and can cause irreversible vision loss [2]. Progression to vision impairment can be slowed or averted if DR is detected in time, however this can be difficult as the disease often shows few symptoms until it is too late to provide effective treatment. Detecting DR is currently a manual, resource intensive and highly time-consuming process. It involves localisation and grading of subtle features like presence of lesions from 7-field colour Fluorescein angiography (fundus) photographs by highly trained clinicians[1]. Computer-automated diagnostics for screening retinal images would definitely reduce costs and allow earlier detection of DR.

## 2. Related Work

Previous work has been done to detect DR automatically using machine learning and statistical models. The methodologies can be categorized as using CNNs and utilising learned features and by extracting features and trying to model features of interests. The authors in [9] for example have used image processing techniques to extract morphological features like lesions and then trained an SVM on a dataset of 330 images and have achieved an accuracy of 85 %.

### 2.1. Using Explicit feature extraction methods

A lot of work has been done in developing image processing techniques to extract features from images of the fundus. But a shortcoming of these studies was that the scope defined was very small and the datasets were homogenous.

### 2.2. Detection of DR using CNN- GoogleNet

Using convolutional neural network to perform classification of fundus images have become a popular method. In [3] the authors have used neural network optimized for image classification and trained using a retrospective development data set of 128,175 retinal images, which were graded 3 to 7 times for diabetic retinopathy, diabetic macular edema, and image gradability by a panel of 54 US licensed ophthalmologists and ophthalmology senior residents between May and December 2015. They have used the Inception v3 architecture, developed by Google, and have fully trained the network from scratch using the fundus images. In this study, sensitivities of 97.5% and 96.1% were achieved.

### 2.3. Detection of DR using CNN - LeNet

In Yang et al[8]. , they have used a CNN (LeNet-5 architecture) as a feature extractor for addressing blood vessel segmentation. The model has three heads at different layers of the convnet which then feed into three random forests. The final classifier uses and ensemble of the random forests for a final prediction achieving an accuracy and AUC on 0.97/0.94 on the DRIVE dataset (a standard dataset for comparing models addressing vessel segmentation).

## 3. Dataset

### 3.1. Overview

Data was drawn from a dataset maintained by EyePacs, and provided via Kaggle. The dataset is composed of color photographs that varied in height and width between the low hundreds to low thousands of pixels. Each image was labeled by a trained clinician and assigned 2 labels - 1 for

presence of DR and 0 for no DR. Below are examples of dataset images:

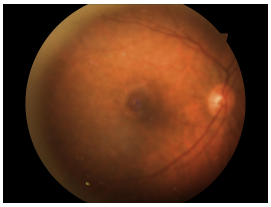
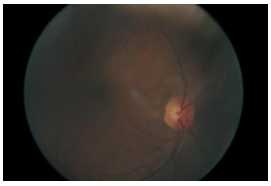


Figure 1: (a) No DR

(b) DR

### 3.2. Heterogeneity and Noise

There are numerous challenges presented by this task and dataset. The fundus photographs present is highly heterogeneous because the images in the dataset are taken from different models and types of camera under varied imaging conditions. The photographs have different resolutions ranging from 2592x1944 to 4752x3168, having vastly different degrees of noise and lighting. Figure 2 shows some examples of the poor quality of images in our dataset.



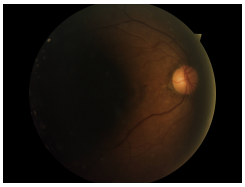


Figure 3: Poor quality images with low lightening

### 3.3. Data Preprocessing

Due to the presence of heterogeneity and noise in our dataset, we performed some preprocessing tasks on the images. While manually analysing these images, we found most of the images contained an excess of insignificant black border that did not provide any information on either side of the image. These black images were cropped as our first task of preprocessing. To deal with varied sized images, we downscaled the images by a factor of 2 and reshaped them to 256 x 256 x 3.

The downscaled and cropped input images were then augmented in real-time by:
1. Cropping with certain probability
2. Colour balance adjustment
3. Brightness adjustment
4. Contrast adjustment
5. Flipping images with 50% chance
6. Rotating images by x degrees, with x an integer in [0, 360].

Augmentation was performed at each epoch, real time by performing random transformations mentioned above on the given batch size of dataset. The images in the dataset were also normalized in the end.

## 4. Approach

We begin by trying to solve the binary classification task on our noisy dataset. We denoise, normalize, and augment the data as described in the preprocessing section. We trained two models, one based on the vgg19 architecture and the other on the inception v3 architecture. We removed the last softmax layer and added a linear layer so as to perform binary classification.

We initially started with training just the last classification layer and using the CNNs as fixed feature extractors. But we realised that since our dataset comprised of fundus images , the CNNs were not able to detect subtle features like micro-aneurysms. So instead we finetuned the CNN using the pretrained weights. For both our models we have used binary cross entropy loss as our loss function.

### 4.1. Class Imbalance

As mentioned above we had a huge class imbalance in our dataset as the number of negative samples being larger than positive samples. To address this problem we used two approaches. First, we trained the models by using a dataset which had down-sampled majority class. In the second method we up-sampled the minority class using data augmentation techniques in real-time. We performed operations like rotation and flip on the minority class and augmented the samples.

### 4.2. VGG19

The first CNN we experimented with was VGG19, built by Karen Simonyan and Andrew Zisserman at Oxford University. We initially started using this model by loading the learned weights and retraining only the last classification layer to give binary classification. Then we decided to finetune the entire network using the learned weights as initializations. This usage of transfer learning is viable because many of the early layers of the network learn similar features, such as edges and lines. By loading these pre-trained weights, our model effectively already knows how to detect lines and edges, and need only learn how to use them to make predictions for our problem.

### 4.3. Inception V 3

The second pre-trained model we used was inception v3, which was developed at Google. Inception v3 has a deeper architecture and we can use the two different heads of the output for more experimentation. Similar to VGG19, we loaded the pre-trained weights into our network, and retrained the final layer to predict 2 classes rather than 1000.

# 5. Experiments

## 5.1. Dataset

We divided our dataset into training and validation dataset, which consisted of a total of 2062 retina images, out of which close to 1500 were negative examples and 500 were positive examples. So our training dataset had 1016 negative and 484 positive samples and validation set had 451 negative and 111 positive examples.

In-order to feed the data into VGG19 the images were resized to 224 by 224 as that is the expected input data size of the VGG network. For the inception v3, we had to resize the images to 299 by 299. We had initially done no pre-processing on the dataset but that was giving bad performance and was a computational overload on our systems. Later we performed the pre-processing of the dataset and that improved the results.

## 5.2. Learning

We used SGD with nesterov momentum for VGG19 and used a learning rate of 0.001. We started by experimenting with the optimal learning rate as is given for for the architecture and explored on learning rates above and below the range, but that caused a decrease in performance. So we fixed on the above setting.

For inception v3, we saw that the initial loss wasn't decreasing much after the first epoch with SGD, so instead we tried ADAM with the optimal settings. It caused a slight boost in the performance.

For both the models we used a learning rate scheduler, where we decayed the learning rate by a factor of 1e-5 after every 2 epochs. We experimented with different values for this setting as well and fixed on the above settings.

The models were trained over 100 epochs with a batch size of 50. Due to the limitation of computational resource, we could not experiment with increasing epochs and batch size over 100 and 50 respectively.

## 5.3. Results

Our models mainly ran on personal Linux machines. Some of the experiments like fine-tuning of the model were performed on Google Cloud compute machines with 8 Intel Xeon E5-2670 (Sandy Bridge) processors and one GPU. Both the VGG-19 and GoogleNet models were developed using Pytorch.

## 5.4. Interpreting results

To evaluate the performance of models on the test set, we are using three performance metrics. First we evaluated our model's performance on accuracy. Since accuracy measures the proportion of examples that were classified correctly, it will be an unfair measure for a dataset that is biased on some class. As our dataset is imbalanced, examples of no symptoms retina images being more than symptoms class, we are additionally using following metrics to evaluate the performance of our models. Recall measures the proportion of positives that were correctly predicted. In medical terms, recall is most commonly known as sensitivity. Sensitivity becomes an important metric to track because it refers to the medical test's ability to correctly detect ill patients who do have the condition.

## 5.5. VGG19

The first model we tried was VGG19. We were able to achieve a training accuracy of 74 % and sensitivity of 68.5 % so we were clearly able to overfit our data. Interestingly, even as we continued to overfit more and more, our validation accuracy and sensitivity remained relatively constant. The best validation and training accuracy and sensitivity for VGG19 is given below along with the confusion matrix.

| Dataset | Training | Validation |
|---|---|---|
| Accuracy | 76% | 74.7 % |
| Sensitivity | 77% | 72% |

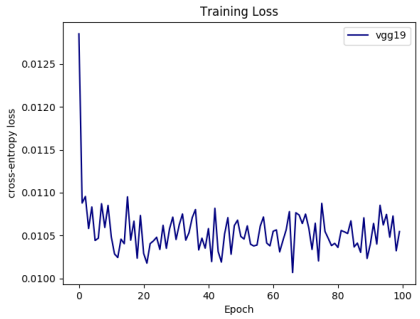



Figure 4: Training loss for VGG19

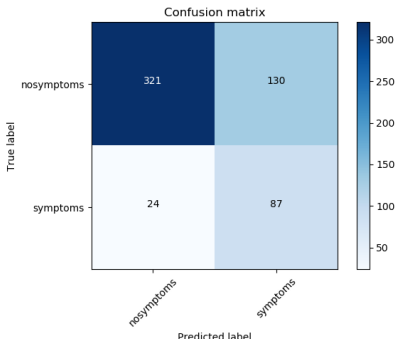



Figure 5: Confusion matrix for VGG19

### 5.6. Inception V3

The second model on which we attempted to use transfer learning was Inception v3, where we had a validation accuracy of 72 % and sensitivity of 65% a Similar to VGG19,our training accuracy was higher than validation accuracy, indicating that we were overfitting our training data. The model did not perform as well as the VGG 19 problem, and we think it could have performed better if we trained it over more epochs, and tried a different hyper-parameter search technique. But unfortunately due to machine constraints we were not able to do so. The best validation and training accuracy and sensitivity for inception is given below.

| Dataset | Training | Validation |
|---|---|---|
| Accuracy | 74% | 73% |
| Sensitivity | 64% | 44% |

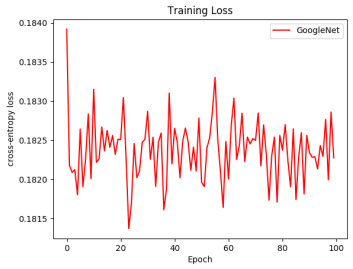



Figure 6: Training loss for Inception V3

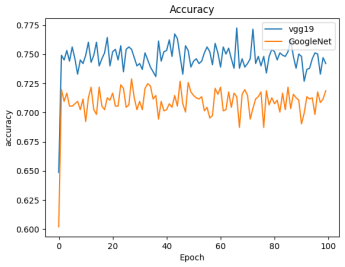



Figure 7: Training accuracy for VGG19 and Inception

### 5.7. Result discussion

As we can see VGG19 performed better than inception v3 for binary classification by a margin of 1 - 2 %. But for sensitivity VGG19 performed much better, for both training and validation. That means that it was able to detect correct DR diagnosis more accurately than the other model.

## 6. Future Works

In the future works, there are few things that we would like to do. First, we would like to try our experiments with more number of datasets. We believe this would help CNNs train better and a give a better accuracy. Second, we would like to modify the network in a manner that it can incorporate an understanding of a patient's both eyes - left and right. This would act as a feature that we believe will help the network to improve its performance.

Additionally, we want to extend the problem of detecting DR from presence of the symptoms to severity of the disease. Thus, training the classifier to be able to detect the severity of DR - mild, moderate or severe. The major issue with such classification is the presence of balanced dataset.

## 7. Conclusion

We presented analysis of two models which we trained for classification of fundus images to detect diabetic retinopathy. The model performs well in comparison to human evaluation metrics. Further work can be done by applying different methods for data pre-processing like normalisation and noise removal. For now the model is detecting the presence of DR in the images, for future work we can perform a multiclass classification to detect the severity of the disease.

## 8. References

[1] Grading diabetic retinopathy from stereoscopic color fundus photographsan extension of the modified airlie house classification: Report number 10.

[2] Linda Geiss et al. Engelgau, Michael. The evolving diabetes burden in the united states. Annals of Internal Medicine, 2004.

[3] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA, 316(22):24022410, 2016.

[4] B. Graham. Kaggle diabetic retinopathy detection competition report. 2015.

[5] Bhat P. S. Acharya U. R. Lim C. M. Nayak, J. and M Kagathi. Automated identification of different stages of diabetic retinopathy using digital fundus images

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going

deeper with convolutions, 2014.

[7] Karen Simonyan Andrew Zisserman. Very deep convolutional networks for large scale image recognition. 2015

[8] G. Cao B. Wei Y. Zheng G. Yang S. Yang, Y. Yin. Hierarchical retinal blood vessel segmentation based on feature and ensemble learning. Journal on Neurocomputing Vol 149, p. 708-717, 2015.

[9] Ng EY Chee C Tamura T. Acharya U, Lim CM. Computer-based detection of diabetes retinopathy stages using digital fundus images. Proceedings of the Institute of Mechanical Engineers, 545-553, 2009.