Abstract

Diabetes is a chronic condition affecting millions of people worldwide. One of its major complications is diabetic retinopathy (DR), which is the most common cause of legal blindness in the developed world. Early screening and treatment of DR prevents vision deterioration, however the recommendation of yearly screening is often not being met. Mobile screening centres can increasing DR screening, however they are time and resource intensive because a clinician is required to process the images. This process can be improved through computer aided diagnosis, such as by integrating automated screening on smartphones. Here we explore the use of a SqueezeNet-based deep network trained on a fundus image dataset composed of over 88,000 retinal images for the purpose of computer aided screening for diabetic retinopathy. The results of this neural network validated the viability of conducting automated mobile screening of diabetic retinopathy, such as on a smartphone platform.

1 Introduction

The number of people living with diabetes is dramatically increasing. This epidemic is observed internationally, with global prevalence of diabetes projected to climb to over 500 million by 2030 [1]. Common health complications faced by patients with diabetes include cardiovascular failure, amputation, nephropathy, and blindness [2]. The latter complication of blindness is caused by diabetic retinopathy (DR), which is the most common cause of legal blindness in the working-age population in the developed world estimated to affect over 93 million people [3]. Currently, screening and monitoring for DR is a time-consuming and manual process requiring a trained clinician to evaluate digital colored photographs of the back of the eye (retinal fundus images) to indicate the presence and stage of retinopathy. Automated image processing may reduce or eliminate the need for human grading, decrease the costs of screening, and increase its accessibility [4].

DR can be identified by the presence of lesions associated with the vascular abnormalities caused by the disease. This is conducted by a clinician and can be extremely resource intensive. The use of digital systems have been found to be a feasible option for mobile screening environment [5], and digital photographic imaging paved the way for future development of retinal fundus image processing. The detection of abnormal features in the eye such as exudates, microaneurysms, and haemorrhages using a trained neural network have been highly effective, and can reduce a third of the workload of a human screener, but it shows little efficacy in clinical practice as a replacement for human screening [4]. In 2015, Kaggle hosted a data science competition for identifying signs of diabetic retinopathy1. The goal of the competition was to push the development of automated detection systems that ideally would have clinical potential.

The aim of this research is to automate screening of retinal fundus images in grading their DR severity levels. The ultimate goal is to implement such an algorithm that could be deployed on a smartphone mobile device to provide full accessibility for diabetic patients to screen their eyes in the comfort of their own home. We implement a compact deep neural network using the principles of SqueezeNet [6], and apply it to the dataset from the Kaggle competition.

1.1 Fundus Image Grading

The images provided in the Kaggle dataset were graded by clinicians according to a five-stage modified Walsh scale [7]. The first stage indicates no diabetic retinopathy, the second to the fourth indicates varying levels of diabetic retinopathy (from mild to moderate to severe), and the fifth stage indicates advanced eye disease. For the purpose of screening, the database provided by Kaggle only includes the first four stages. See Figure 1 for examples of fundus images of severity 0 to 4.

Fig. 1: Example of retinal fundus images of severity level 0 to 4 (shown in top left corners).

1.2 Algorithms for Diabetic Retinopathy Screening

For screening retinal fundus images, one approach has been to use algorithms to segment retinal vessels. Retinal morphology can be applied to assessing eye diseases, including Glaucoma and Diabetes. CNNs have been applied to this problem, and achieved high levels of accuracy [8, 9, 10]. With regards to direct DR severity level classification, a number of efforts have been made using deep learning. In [11], a relatively shallow neural network was trained to classify 5 severity levels with high accuracies. In [12], a variant of the Inception-v3 [13] network, was trained to grade fundus images for diabetic retinopathy, and diabetic macular edema. The model was shown to perform well in both tasks. In [14], the use of a Weighted Kappa [15] loss function was explored, which can help incorporate information about class distributions into training.

In the 2015 Diabetic Retinopathy Kaggle Competition, all of the top teams in the competition implemented CNNs in some form. Among the top performers, the typical strategy was to use ensemble networks, where multiple large deep networks were trained and combine at test time [16, 17, 18]. The winner of the competition utilized three separate models to form their overall system, along with a random forest to post process the model outputs [16].

---

1. The dataset used was available at <https://www.kaggle.com/c/diabetic-retinopathy-detection>.
Table 1: Network Architecture. All convolutional layers utilize a ReLU activation.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Shape/ Details</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td>512x512x3</td>
<td>512x512x3</td>
</tr>
<tr>
<td>Conv1</td>
<td>7x7x32 stride 2</td>
<td>256x256x32</td>
</tr>
<tr>
<td>Pool1 (Max)</td>
<td>3x3 stride 2</td>
<td>128x128x32</td>
</tr>
<tr>
<td>Fire1</td>
<td>1x1x16 → 1x1x64, 3x3x64</td>
<td>64x64x128</td>
</tr>
<tr>
<td>Fire2</td>
<td>1x1x16 → 1x1x64, 3x3x64</td>
<td>64x64x128</td>
</tr>
<tr>
<td>Bypass1</td>
<td>Fire1 + Fire2</td>
<td>64x64x128</td>
</tr>
<tr>
<td>Fire3</td>
<td>1x1x32 → 1x1x128, 3x3x128</td>
<td>64x64x256</td>
</tr>
<tr>
<td>Pool2 (Max)</td>
<td>3x3 stride 2</td>
<td>32x32x256</td>
</tr>
<tr>
<td>Fire4</td>
<td>1x1x32 → 1x1x128, 3x3x128</td>
<td>32x32x256</td>
</tr>
<tr>
<td>Bypass2</td>
<td>Pool2 + Fire4</td>
<td>32x32x256</td>
</tr>
<tr>
<td>Fire5</td>
<td>1x1x48 → 1x1x192, 3x3x192</td>
<td>32x32x384</td>
</tr>
<tr>
<td>Fire6</td>
<td>1x1x48 → 1x1x192, 3x3x192</td>
<td>32x32x384</td>
</tr>
<tr>
<td>Bypass3</td>
<td>Fire5 + Fire6</td>
<td>32x32x384</td>
</tr>
<tr>
<td>Fire7</td>
<td>1x1x64 → 1x1x256, 3x3x256</td>
<td>32x32x512</td>
</tr>
<tr>
<td>Pool3 (Max)</td>
<td>3x3 stride 2</td>
<td>16x16x512</td>
</tr>
<tr>
<td>Fire8</td>
<td>1x1x64 → 1x1x256, 3x3x256</td>
<td>16x16x512</td>
</tr>
<tr>
<td>Bypass4</td>
<td>Pool3 + Fire8</td>
<td>16x16x512</td>
</tr>
<tr>
<td>Dropout</td>
<td>Keep probability 0.5</td>
<td>16x16x512</td>
</tr>
<tr>
<td>Pool4 (Max)</td>
<td>3x3 stride 2</td>
<td>8x8x512</td>
</tr>
<tr>
<td>Conv2</td>
<td>1x1x64</td>
<td>8x8x64</td>
</tr>
<tr>
<td>Flattening</td>
<td>3D to 1D</td>
<td>4096x1</td>
</tr>
<tr>
<td>FC1</td>
<td>4096x5</td>
<td>5x1</td>
</tr>
<tr>
<td>Softmax</td>
<td>5x5</td>
<td>5x1</td>
</tr>
</tbody>
</table>

In the majority of these approaches however, large models are trained to perform operations. While large networks provide great results, they are not always the most practical to deploy, especially for mobile screenings.

2 Methods

Deep learning can be a very powerful tool for many image related problems. When it comes to creating the best performing networks, often the strategy is to train very large networks with millions of parameters, however, these networks can be cumbersome to run and deploy. Our goal is to create a more compact network that can be more feasible in deployment, even on a smartphone mobile platform.

In order to create a compact network, we utilized many of the design principles from SqueezeNet [6]. SqueezeNet utilizes a number of strategies to reduce the number of network parameters, while trying to still maintain high levels of performances. Our network architecture incorporates their Fire modules and simple layer bypasses. The network layout is shown in Table 1. Dropout [19] was used after the last Fire module. Our network has a total of 807,013 trainable parameters, and can be saved in a file size of 9.7 MB without using any additional model compression techniques.

2.1 Dataset

We used the 2015 Diabetic Retinopathy Detection Kaggle Competition dataset, which was provided by EyePACS2 and consists of 35,126 training retinal fundus images and 53,576 test images. Each image was rated by clinicians for the presence of diabetic retinopathy, using severity levels from 0 to 4. The images are a collected from different types of cameras, often with various types of artifacts present.

The fundus images in the database contain different borders, illuminations levels, contrast, and are also inconsistent in their placement in the image. In order to provide more consistency in the training set, the black border around the fundus was segmented and cropped out using Otsu’s method [20]. The amount of black border cropped is unique for each image. Only the information sections of the fundus is kept. Each image is padded with zeros to have equal dimensions and is scaled down to 512 by 512. We additionally apply contract enhancement to each image [16]. Figure 2 shows the results of our pre-processing. During training, random combinations of rotation, flipping, scaling and cropping were used to augment images.

2.2 Training Details

Our network was implemented inTensorflow3 on a Linux machine with an Nvidia Titan X GPU. Training was done with mini batch sizes of 128 using the Adam optimizer [21] and a learning rate of 0.0001. For our final model, training was done over about 550 epochs, and took approximately 70 hours to complete.

Fig. 2: Image preprocessing. Top: Original image. Bottom: Pre-processed image.

3 Results

Our overall accuracy on the test set was 83.6%. At test time, we run 5 crops of each test image through the network and average the model outputs to produce our overall classification. The crops we used were 10% crops positioned at each of the corners and one 10% crop of the center of the image. This was done to improve the invariability of the network, which provided a slight performance increase.

Rankings in the Kaggle competition were not done based strictly on the accuracy of a team’s submission. It was instead based on a calculation of Quadratic Weighted Kappa score [15]. This score is meant to represent the level of agreement between two sets of classifications, while accounting for the likelihood of agreement by chance. This score does not necessarily increase with the model accuracy, since the Kappa score accounts for the distributions of classifications. In our case, our best scoring model did not also produce our highest accuracy on the test set. The winning team, Min-Pooling [16], achieved a submission Kappa score of 0.84958. Our Kappa score was lower but still strong at 0.76543, and would have given us a rank of 20.

2eye pals.com
3tensorflow.org
4 Discussions and Conclusions

We focused on utilizing a compact network in our system, however our performance could likely be improved by using a larger diagnostic system. The majority of the top teams incorporated more complex systems with additional components. Many teams utilized very large networks, with many times more parameters [16, 17, 18]. A common strategy employed by the top teams was also to utilize an ensemble of networks working together. By using multiple networks with different layouts, the overall system can be made more robust. The third place team [18] in particular, utilized nine large CNNs, computing on different resolutions, in combination.

Patient eye blending as a post processing step was also a common strategy among the teams. This strategy uses the initial predicted severity levels of each eye to jointly refine the predictions of both eyes [16, 17, 18]. Additionally, some teams also incorporated some post processing steps designed specifically to maximize the calculated Kappa score [16, 18].

We were able to show that using a compact network alone is capable of adequately conducting screening, resulting in a strong score of classification. Our system was significantly smaller and less complex than that of the top teams, making it more practical for deployment. The use of SqueezeNet demonstrate the viability of using a relatively small network to achieve strong classification scores.

Future research is still needed to be done to assess the efficacy of this system in a clinical setting and to assess its application when deployed to alternative platforms such as a smartphone device. This latter use-case would be advantageous in increasing patient compliance and patient education regarding DR.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

References


