Machine Learning Engineer Nanodegree

Capstone Project

Madalina Buzau
March 21st, 2017

I. Definition

Project Overview

Diabetic retinopathy (DR) is one of the major causes of blindness in the entire world, approximately 93 million people are affected to various degrees by it [1]. Usually, DR detection is done manually by trained clinicians being a very time consuming process which often leads to delays in both diagnosis and treatment [2]. Early detection of DR can slow down the visual decline caused by it thus it is really important to develop fast, accurate and reliable algorithms for its detection.
On a personal note, delaying visual decline is a subject very close to my heart. Both of my grand-parents lost their vision to some extent. Though preventive measures are too late for them, I would be really happy to know that I will be working on an issue that could potentially help other people keep their gift of sight for as long as possible.
In this project, we will develop a machine learning model that is able to rate the presence of DR in someone's retina. The data used for training and testing the performance of the model comes from the Diabetic Retinopathy Detection challenge held on Kaggle.

Problem Statement

The goal of this problem is to develop an algorithm that is able to rate the presence of DR in someone's retina image. The rating is done on a scale of 0 to 4, being defined as follows [3],[4]:

- 0 - No DR. No abnormalities present.
- 1 - Mild. Microaneurysms only.
- 2 - Moderate. More than just microaneurysms.
- 3 - Severe. Any of the following: >20 intraretinal hemorrhages in each of 4 quadrants, definite venous beading in 2+ quadrants, prominent intraretinal microvascular abnormalities (IRMA) in 1+ quadrant.
- 4 - Proliferative DR. Either neovascularization or vitreous/preretinal hemorrhage. Or both of them.

To develop the algorithm, the following steps will be taken:

- Download the training dataset from the website of the competition
- Preprocess the high-resolution retina scans
- Create a training, validation and testing dataset
- Perform data augmentation on the training dataset
- Train a classifier on the training dataset and tune the hyperparameters using the validation dataset
- Check the final performance of the classifier on the test dataset.

This trained classifier can be afterwards used to rate the presence of DR on new retina scans.

Metrics

For evaluation purposes we will use the metric used by Kaggle on this competition: the quadratic weighted kappa (QWK) score. This will be helpful for comparing the performance of our model with the benchmark.
The QWK score measures the agreement between two raters whilst taking into account the agreement that can occur by chance [5]. In our case, the raters are the ground truth labels provided by clinicians (rater A) and the labels predicted by the classifier (rater B).
A negative QWK value indicates that the classifier performs worse than a naive classifier which predicts the target randomly. A QWK score of 1 indicates that the algorithm has perfect predictions. The QWK score is defined as follows [6]:

$$QWK = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}$$

where $O_{i,j}$ is the matrix of the observed scores and $E_{i,j}$ is the matrix of the expected scores if we would be predicting randomly and

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

and represents a weight matrix that indicates the seriousness of disagreement between two raters [5].

II. Analysis

Data Exploration

The data used to train, validate and test the model for DR rate detection comes from the train dataset hosted on Kaggle. The images in this dataset consist of

high-resolution retina scans on each eye of the patient and come with the DR rate manually annotated by clinicians. As each patient has two retina scans, the total number of patients in the train dataset will be $N_{patients} = \frac{N_{images}}{2}$. Table 1 shows the number of images in train dataset.

Table 1: Train and test data

Dataset	Number of images	Image format
Train	35126	JPEG

These images have really high resolution making it very difficult to run experiments with image preprocessing and model tuning. Thus, we will be working on a smaller dataset.
As it can be seen in Table 2, the classes in the original dataset are very imbalanced. This is however expected. Severe cases with 3 and 4 DR rate are not as often encountered as the mild DR cases. It is actually a relief to see this!
To mitigate the imbalance between classes, the images in the reduced dataset were chosen in such a way that the data imbalance is reduced. We can consider this selection as an undersampling of the majority classes.
Below, you can find the distribution of the targets in the original and reduced dataset.
In the reduced dataset, the DR rate 3 and 4 are still minority classes but now

Table 2: Distribution labels in the original and reduced dataset

DR rate	Original number of samples	Reduced number of samples	Original percentage	Reduced percentage
0	25810	2000	73.4783%	26.38%
1	2443	2000	6.9549%	26.38%
2	5292	2000	15.06%	26.38%
3	873	873	2.4853%	11.51%
4	708	708	2.0156%	9.33%

their percentage is reasonably higher.

Exploratory Visualization

For data visualization, I believe it is important to see some cases with and without DR detected. As it can be seen in Figure 1, if you are not a trained clinician it is really difficult to distinguish between classes. It is certainly not an easy task for a machine learning algorithm.
We can also observe that the resolution of these retina scans is extremely high. This can be considered as a plus, as a machine learning algorithm will have access to the finest details of the retina scan. However, depending on the type

of the algorithm chosen, it can be extremely difficult even impossible to train it on such large images.

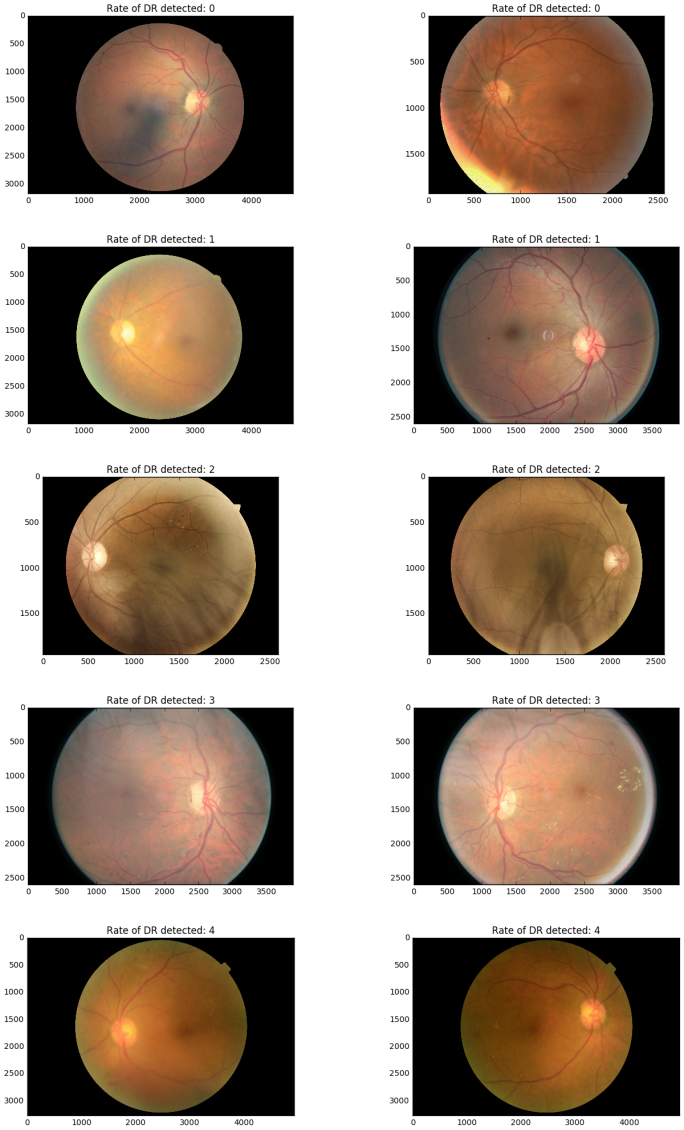

Figure 1: Cases with various DR detected

Algorithms and Techniques

To select the hyperparameters of the model and to evaluate its performance, the dataset has been split in a training, validation and test dataset.
As the input dataset consists of images, we will attempt to solve this detection task using Convolutional Neural Networks (CNNs). Deep CNNs have proven to work really well on image classification tasks [7]. Unlike other machine learning models, a CNN will preserve the spatial structure of the image. The network will use the retina image as an input and will attempt to create its own set of features through the use of feature maps in order to predict as accurately as it can the targets.
Figure 2 shows the main structure of a CNN. A CNN works with three dimensions: width, height and depth (number of color channels, 3 in our case). The input layer in our case is the retina scan whilst the following layers are convolutions that go through each patch of the previous layer to compute non-linear activations.

There are a lot of techniques that can be used to improve the performance

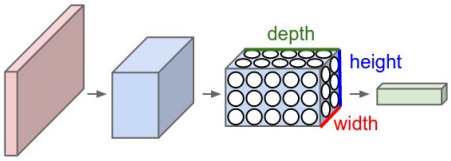

Figure 2: CNN structure (Source [7])

of CNN and also a lot of hyperparameters to tune. The most important hyperparameters that can affect significantly the performance of the algorithm are:

- the learning rate - controls the step of the weights update. If it's too small, it will take longer to get to the local minima. If it's too big, there's the risk of overshooting it.
- the kernel size of the convolutions
- the activation function (e.g. sigmoid, ReLU, ELU)
- the number of layers, feature maps

Training CNN from scratch is extremely difficult. It is very computationally expensive and chances are that you will have to tune the hyperparameters a lot to get decent results. To offer us a good start, we will be using **transfer learning**.
With this method, the weights of a pre-trained network are used as a starting point during training. For our model, we will be using the pre-trained weights of the deep residual networks ResNet-50 [8] which won both COCO and ImageNet challenges in 2015. The weights were trained for the ImageNet classification

task and the fully connected layer was removed from the model. Two new fully-connected layers were added on top of the model.
To avoid the risk of overfitting, we will also be using 'early stopping' [9] on the validation dataset. With this technique, the training is stopped if there is no further improvement on the validation dataset. Dropout before the first fully connected layer has also been used to mitigate the risk of overfitting [10].
Another important aspect is the cost function we choose when training. As the metric for this competition was the QWK score, it would be ideally if during training we would optimize this metric. However, the QWK function is non-differentiable thus it cannot be optimized with gradient descent. The cost function used for training our model was the Softmax cross-entropy cost function

$$L = -log(S_y)$$

which takes the predicted probability for the correct class

$$S_y = \frac{e^{f_y}}{\sum_{i=1}^{C} e^{f_i}}$$

and passes it through the natural logarithm function. Batch normalization [11] has also been used to reduce the risk of internal covariate shift and make the network less sensitive to weights initialization. To increase the accuracy on unseen samples, data augmentation techniques have been used on the training dataset.

Benchmark

As the competition has already closed, I will consider as benchmark the winning model. The best model for this competition achieved a ~0.85 quadratic weighted kappa score on the leaderboard, using sparse CNN as a first stage classifier and then random forests as a second stage classifier [12].

III. Methodology

Data Preprocessing

One of the most important steps during model development was data preprocessing. Most of the retina scans were surrounded by a black border like in Figure 3. This black border varied in size for each image thus it was very difficult to remove it with precision. The best results were obtained by using the **findContours** function from OpenCV. After finding all the contours in the image, a bounding box has been used to crop the largest object found. In our case the largest object is precisely the retina. Figure 4 shows some before and after examples. Due to computational constraints the resolution of the image has been reduced to 270x270px. Smaller resolutions have also been tried but

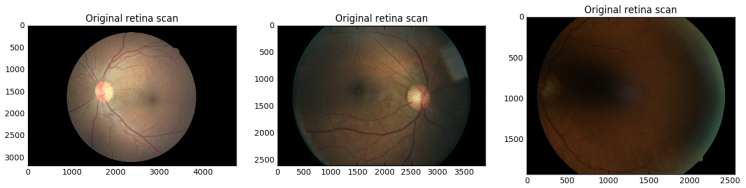

Figure 3: Untrimmed original retina scans

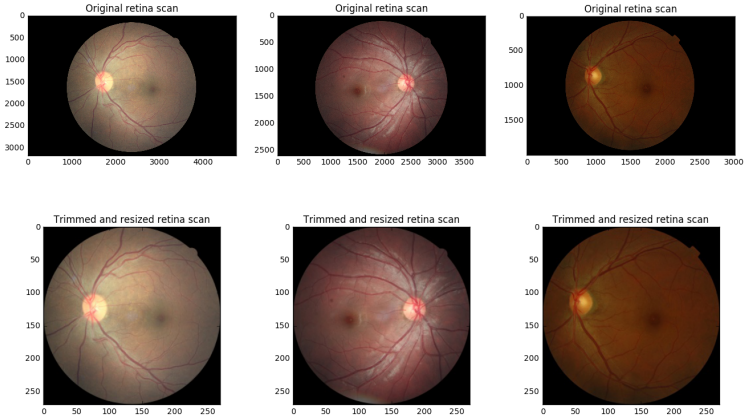

Figure 4: Original vs. Trimmed and resized retina scans

they achieved significantly worse results. Last but not least, all the pixels in the images have been normalized between [0,1].

Implementation

To train the model a training dataset has been created using 80% of the original dataset. The rest of 20% has been split equally into a validation and testing dataset. In Table 3, the distribution of samples in each dataset is shown.

Table 3: Distribution of samples in the training, validation and testing dataset

DR rate	Train No. Samples	Validation No. Samples	Test No. Samples
0	1601	200	200
1	1585	206	209
2	1611	194	195
3	698	91	84
4	569	68	71

To build and train the CNN, Keras with the TensorFlow backend has been used. The ResNet-50 is already available in Keras so it was just a matter of downloading it. The pre-trained weights have been used as a sort of weight initialization. The ResNet-50 was included in the backpropagation during gradient descent in order to tune the convolutions better for our specific task. A dropout layer with a probability of 0.5 has been added before the first fully connected layer. The entire CNN architecture can be seen in Figure 5.

To implement 'early stopping', a callback has been created which monitors the

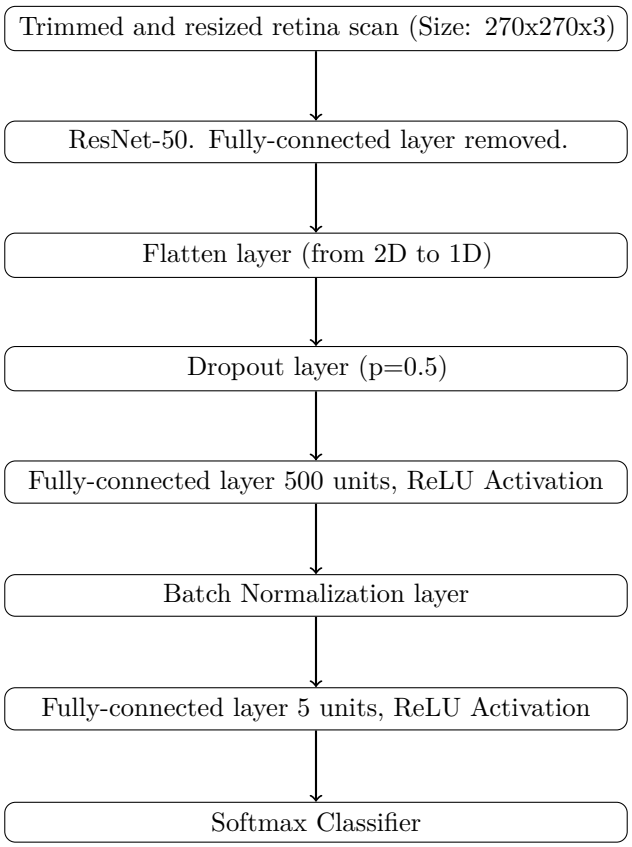

Figure 5: CNN Architecture

accuracy on the validation dataset. If no improvement occurs after 20 epochs the model stops training. Another callback has been created which saves the model that obtained the best accuracy on the validation dataset. The model has been trained with Stochastic Gradient Descent with a learning rate of 0.01 and Nesterov Momentum.

To assess the performance of the model on the validation and test dataset, the **Cohen Kappa Score** from the **scikit-learn** metrics library has been used. The weights have been chosen to be 'quadratic' in order to be compatible with

the benchmark score.
For image augmentation, the **ImageDataGenerator** function from Keras has been used. The following random augmentations have been applied when generating a new batch of images:

- Horizontal flip
- Vertical flip
- Width shift with a range of 5%
- Height shift with a range of 5%
- Shear with a range of 5%

The model has been trained using the **fit_generator** function, which generates 32 augmented images each batch, 250 times per epoch. In the Figure 6, some examples of augmented images are shown.

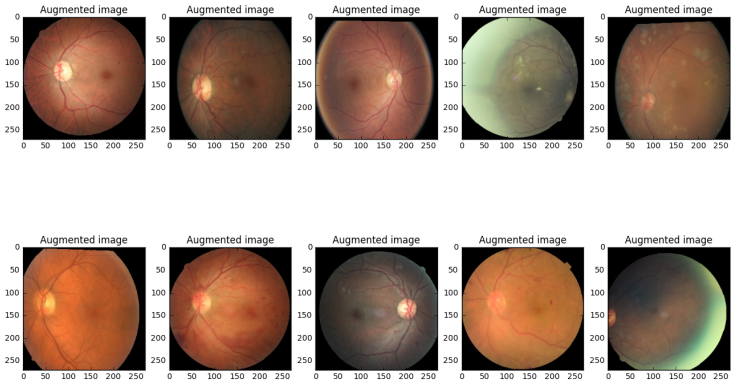

Figure 6: Example augmented images

Refinement

The refinement of the model started right with the data preprocessing step. When downscaling images to 120x120px, the model had a very poor accuracy even on the training dataset. It couldn't get past 30% accuracy, obviously having a very high bias. Once the resolution has been increased to 270x270px the performance of the model has increased dramatically on the training dataset but the performance was still not very high on the validation dataset achieving a maximum accuracy of 29%. As I was using initially a dropout layer with a probability of 0.3 I have decided to increase this probability to 0.5. This made

the model learn slower on the training dataset but it improved the performance on the validation dataset tremendously achieving an increase of 18%.

IV. Results

Model Evaluation and Validation

As mentioned previously, the model has been validated and evaluated using 20% of the reduced training data. The assessment has been done on the QWK scores in order to compare the performance of the model with the winning model of this competition. Figure 7 shows the QWK score obtained on the training, validation and test datasets. As you can see, the QWK score on the training dataset didn't reach 1 as we used 'early stopping' to avoid overfitting. In my previous experiments, without using this technique, the model achieved very easily a 0.99 on the train dataset.
The gap between the QWK score obtained on the validation dataset and test dataset is not significantly high. Thus, the model seems that it doesn't fail to generalize to new unseen samples.

Figure 7: QWK scores on the training and validation datasets

Figures 8, 9, 10 show the confusion matrices obtained on all three datasets. The confusion matrix provides a summary of how well the model assigned the samples to their ground truth label. As you can see in all three matrices, the model is confusing the classes mostly with the adjacent classes. It rarely confuses a normal retina scan with a severe DR scan. I believe that by choosing a higher resolution the model will be able to distinguish better between adjacent classes.
I would consider this model robust enough to be used in the real environment, if it would be trained using the entire training dataset provided by Kaggle. Obviously, the more data available the better will the model generalize to new samples.
The performance on the validation and test dataset leads me to believe that the methodology implemented is correct as there isn't a significant difference between their QWK scores. Last but not least, I believe the performance of this model is very reasonable given the amount of data used and that its structure should be kept when adding more retina scans to the training dataset.

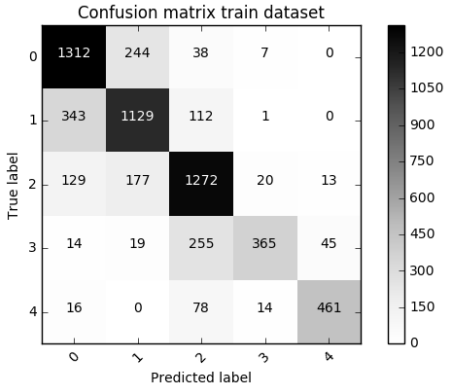

Figure 8: Confusion matrix of the training dataset

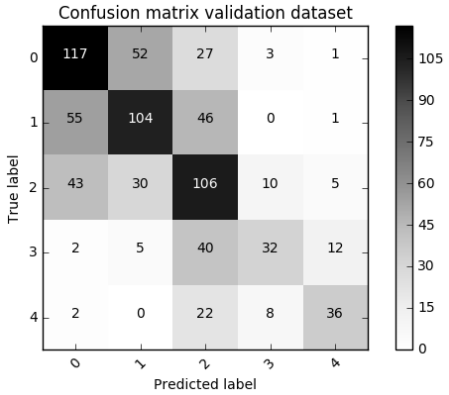

Figure 9: Confusion matrix of the validation dataset

Justification

The winning model achieved a 0.849 QWK score on the test dataset. Nevertheless, the test dataset hosted by the competition was significantly higher. The winning solution was based as well on CNN.
There were 661 teams competing in this challenge. Looking at the leaderboard, the QWK score obtained on the test dataset places us on the 44th place. Assuming the performance of our test dataset is similar to the performance on the entire test dataset hosted by Kaggle, the model is in the 93% percentile.
Given the number of samples that have been used during training, I believe the model manages to learn really well and generalize to new data. There were many techniques that have been tried but were not included in the final version of the model as it didn't improve its performance. Here are some of the techniques that have been experimented with but didn't improve the performance

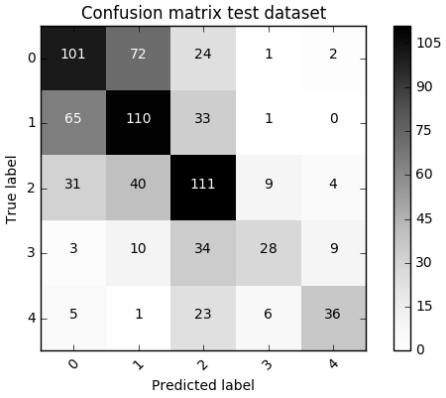

Figure 10: Confusion matrix of the test dataset

of the model:

- Standardizing images with zero mean and unit standard deviation. The validation and test dataset were standardized with the mean and standard deviation of the training dataset.
- Treating the problem as a regression task rather than a classification task. Instead of using a Softmax classifier, a regression layer has been used and trained on a Mean Squared Error cost function. The results obtained using this approach were not as good as using a Softmax classifier.
- Local Contrast Normalization to correct the different illumination in scans. The Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm from OpenCV has been used.
- Custom CNN
- Other pre-trained CNN available in Keras - VGG16, VGG19, InceptionV3, Xception (didn't fit into memory)

V. Conclusion

Free-Form Visualization

In Figure 11, some examples of images from the validation dataset are shown, along with their actual and predicted classes. As it can be observed on the lower part of the figure, the model struggles to identify DR in images that are very poorly illuminated.

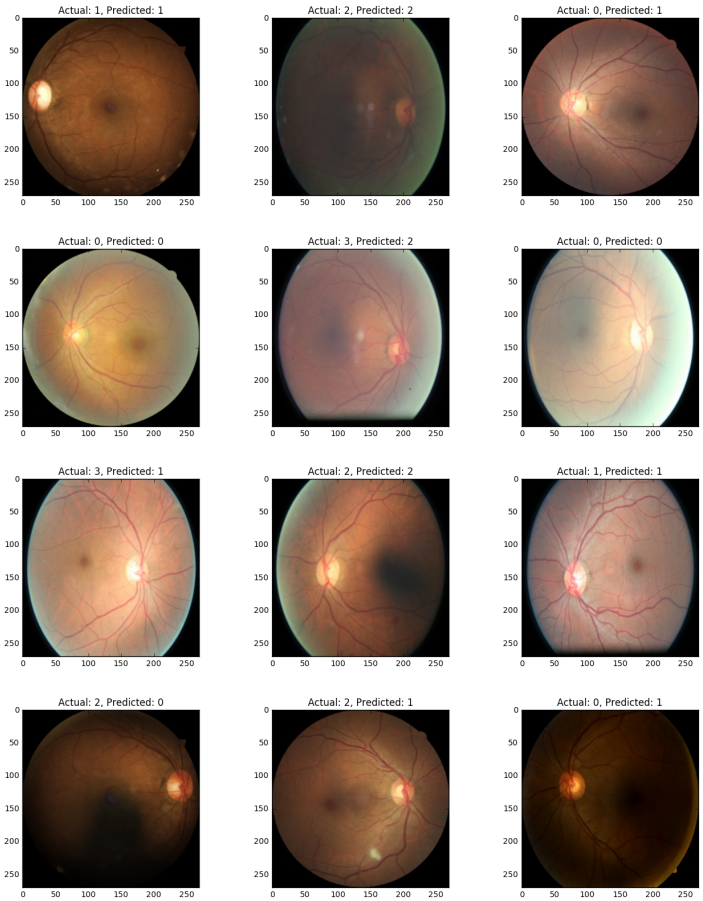

Figure 11: Actual and predicted labels from the validation dataset

Reflection

This report presented a methodology for DR detection in retina scans using CNN.
One of the most important steps in this project was data preprocessing. I have read a few blogposts on this subject (DeepSense, Jeffrey de Fauw) and they all seem to have struggled finding a good way to remove the black border surrounding the images. I believe that my solution worked really well on removing it, after manually inspecting quite a few scans.
The main struggle that I have encountered was working with such large images. It was difficult experimenting with new techniques as each transformation took a long time.
Data augmentation was also very important. I have tried removing this step and the model wasn't able to increase its accuracy on the validation dataset. It simply stopped improving when reaching a certain threshold.
In the beginning, when I have used my own CNN I wasn't even able to load more than 10 images each batch which made the learning process extremely looong and tedious. By switching to ResNet-50 I was able to load 32 images per batch which seemed to improve the learning process tremendously.
The resolution of these images I believe it is very important especially given the nature of this classification problem. Microaneurysms are much easier to spot on with a higher resolution. If I would have had bigger RAM capabilities, I would have definitely chosen larger images.

Improvement

An improvement of the model would be to use the entire training dataset using images with a higher resolution.
Another interesting approach would be to experiment with different color spaces such as the LAB color channels. This could improve the model by finding a better way to represent the retina scans. Maybe a different color space manages to represent the microaneurysms better.
Another possible improvement would be to use attention based models such as Spatial Transformers [13] or Recurrent Neural Networks (RNN) with attention mechanisms [14].

References

[1] Casanova R, Saldana S, Chew EY, Danis RP, Greven CM, Ambrosius WT. Application of Random Forests Methods to Diabetic Retinopathy Classification Analyses. Zheng Y, ed. PLoS ONE. 2014;9(6):e98587. doi:10.1371/journal.pone.0098587.
[2] Diabetic Retinopathy Detection | Description | Kaggle. (2017). [online]

Kaggle.com. Available at: https://www.kaggle.com/c/diabetic-retinopathy-detection#description [Accessed 19 Feb. 2017].
[3] Diabetic Retinopathy Detection | Data | Kaggle. (2017). [online] Kaggle.com. Available at: https://www.kaggle.com/c/diabetic-retinopathy-detection/data [Accessed 19 Feb. 2017].
[4] American Academy of Ophtalmology, (2017). [online] Available at: http://www.icoph.org/dynamic/attachments/resources/diabetic-retinopathy-scale.pdf [Accessed 23 Mar. 2017].
[5] Cohen's Kappa. [online] Available at: https://en.wikipedia.org/wiki/Cohen's_kappa [Accessed 20 Feb. 2017].
[6] Diabetic Retinopathy Detection | Evaluation | Kaggle. [online] Available at: https://www.kaggle.com/c/diabetic-retinopathy- detection#evaluation [Accessed 20 Feb. 2017].
[7] CS231n Convolutional Neural Networks for Visual Recognition. (2017). [online] Cs231n.github.io. Available at: http://cs231n.github.io/ [Accessed 19 Feb. 2017].
[8] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.
[9] Prechelt, Lutz. "Automatic early stopping using cross validation: quantifying the criteria." Neural Networks 11.4 (1998): 761-767.
[10] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.
[11] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).
[12] Kaggle Diabetic Retinopathy Detection Competition Report [online] Available at: https://www.kaggle.com/c/diabetic-retinopathy-detection/ discussion/15801 [Accessed 20 Feb. 2017].
[13] Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." Advances in Neural Information Processing Systems. 2015.
[14] Xu, Kelvin, et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." ICML. Vol. 14. 2015.