Udacity Machine Learning Nanodegree 2019

Capstone Proposal

Diabetic Retinopathy Detection using CNN

Identify signs of diabetic retinopathy in eye images

Atiq Sayyed

I. Definition

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 . 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. 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.

Currently, detecting DR is a time-consuming and manual process that requires a trained clinician to examine and evaluate digital color fundus photographs of the retina. 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. As the number of individuals with diabetes continues to grow, the infrastructure needed to prevent blindness due to DR will become even more insufficient.

On a personal note, after doing the Dog Breed classifier I started loving this Computer Vision area specifically, it's quite intriguing how these amazing Deep Neural Network can learn so much with just forward feed, gradient descent and back propagation. I clearly see this as my area of interest and would also like to grow in this particular field of computer vision.

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 end 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 in- traretinal microvascular abnormalities (IRMA) in 1+ quadrant.
- 4 - Proliferative DR. Either neovascularization or vitreous/preretinal hem- orrhage. 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.

Evaluation Metrics

The evaluation and results of trained models is calculated by common classification metrics which are defined as follows:

- $Precision\ (positive\ predictive\ value) = TP\ /\ (TP+FP)$
- Recall (true positive rate) $= TP\ /\ (TP+FN)$
- $F1$-score $= 2\times TP\ /\ (2\times TP+FP+F)$

Where TP is the number of positive cases which are labelled correctly, TN is the number of negative cases which are labelled correctly, FP is the number of positive cases which are labelled falsely, and FN is the number of negative cases which are labelled falsely. Also since the distribution of the number of samples among database are highly unbalanced, F1-score result which is the harmonic mean of precision and recall, is reported and in order to have a graphical view of the trade-off between sensitivity and specificity metrics, ROC curves and their associated AUC values are used.

II. 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) = N (images)/2

The total dataset was of 80GB and all the images are of high resolution, it was difficult to work, train and evaluate this big dataset, so I picked $1/5^{th}$ part of the data to train and test with the model. Table 1 shows the data available on Kaggle

Table 1: Train and test data

Dataset	Number of Images
Train	35126

Out of 35126, I only used 22129 due to space and other constraints

These images have really high resolution making it very difficult to run experiments with image pre-processing 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!

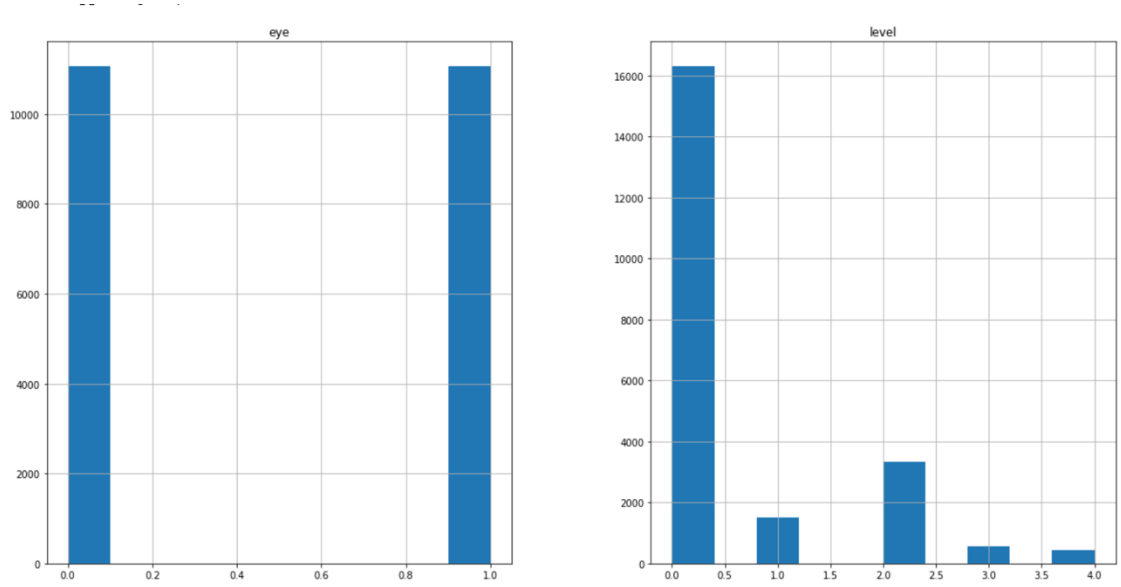
To mitigate the imbalance between classes, the images in the reduced dataset were chosen in such a way that the data imbalance is reduced. We can consider this selection as an under sampling of the majority classes.
Below, you can find the distribution of the targets in the original and reduced dataset.

DR Rate	Original number	Reduced number of samples
0	25810	440
1	2443	440
2	5292	440
3	873	440
4	708	440

Exploratory Visualization

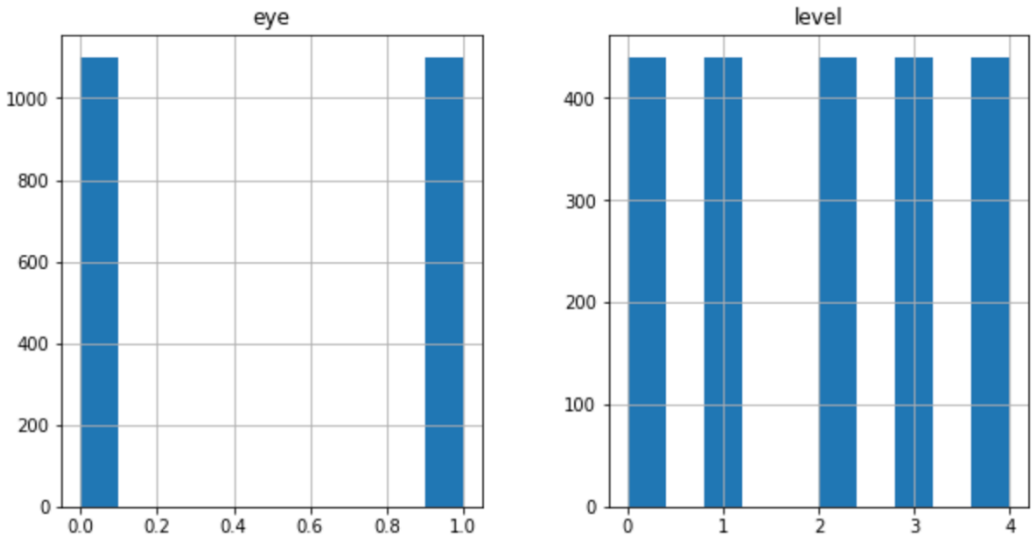
It is clearly visible from below image that the dataset is quite imbalanced. Which was kind of expected there won't be much cases with 3 and 4 DR rate are not encountered as often as the mild DR cases.

Figure 1 Dataset distribution for Eye and it's respective DR rate



To mitigate the imbalance between classes, the images in the reduced dataset were chosen in such a way that the data imbalance is reduced. Refer Figure 2 for even distribution of DR rate

Figure 2 Dataset distribution for even level of DR rate

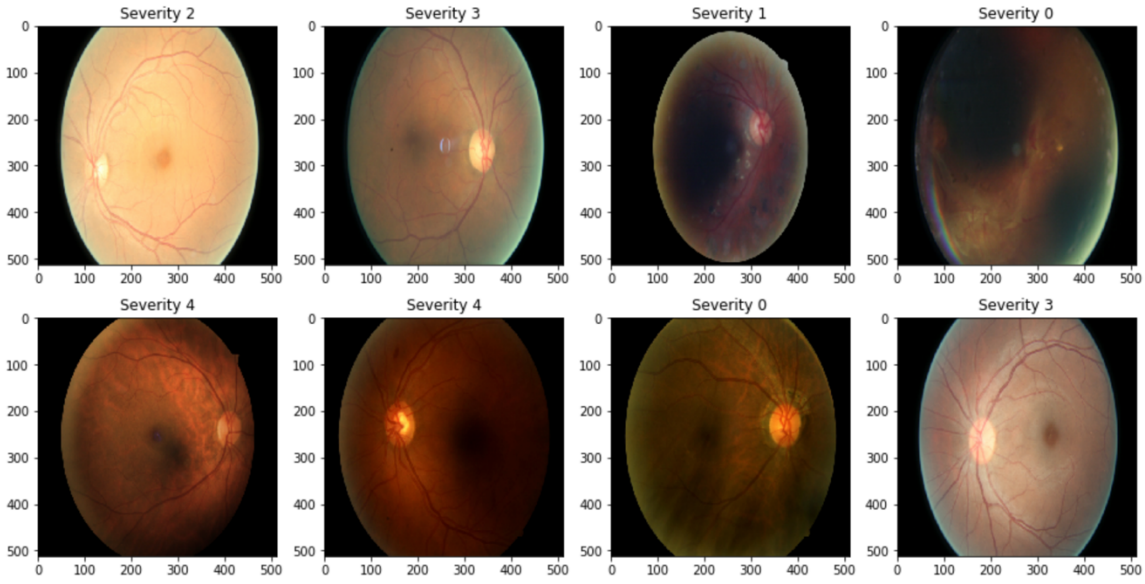


Let's also look at the raw data and how do these images actually look for various severity level, I've picked 8 sample of mixed rate so that we can see the original images of these DR levels.

I believe it is important to see some cases with and without DR detected. As it can be seen in Figure 3, 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, hence I resized them to 512 x 512 for better processing.

Figure 3 Original Images at different severity level



Algorithms and Techniques

The data is divided into training, validation and test data set to evaluate its performance. As the input data is image this is a classical computer vision problem and could easily be tackled with Convolutional Neural Networks (CNNs). Deep CNNs have proven to work really well on image classification tasks. 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.

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 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 Inception-V3. 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' 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. Another important aspect is the cost function we choose when training. As a metric we will use Accuracy, Precision, sensitivity and specificity metrics, ROC curves and their associated AUC values.

Batch normalization 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 score on the leaderboard, using sparse CNN as a first stage classifier and then random forests as a second stage classifier.

III. Methodology

Data Preprocessing

Data Preprocessing was one of the interesting steps in this whole process. The dataset contained images from patients of varying ethnicity, age groups and extremely varied levels of lighting in the fundus photography. The original image has a very high resolution and its extremely difficult to fit in so many images and train the model. Hence I decided to resize the image to 512 x 512 resolution.

There were a lot of other optimization done, while training the data all the training Images went under Augmentation for horizontal and vertical flip, brightness, saturation and contrast refer Figure 6 .

As shown in Figure 4 and Figure 5 the original images have been resized to

Figure 4 Original Image with the actual resolution and Severity

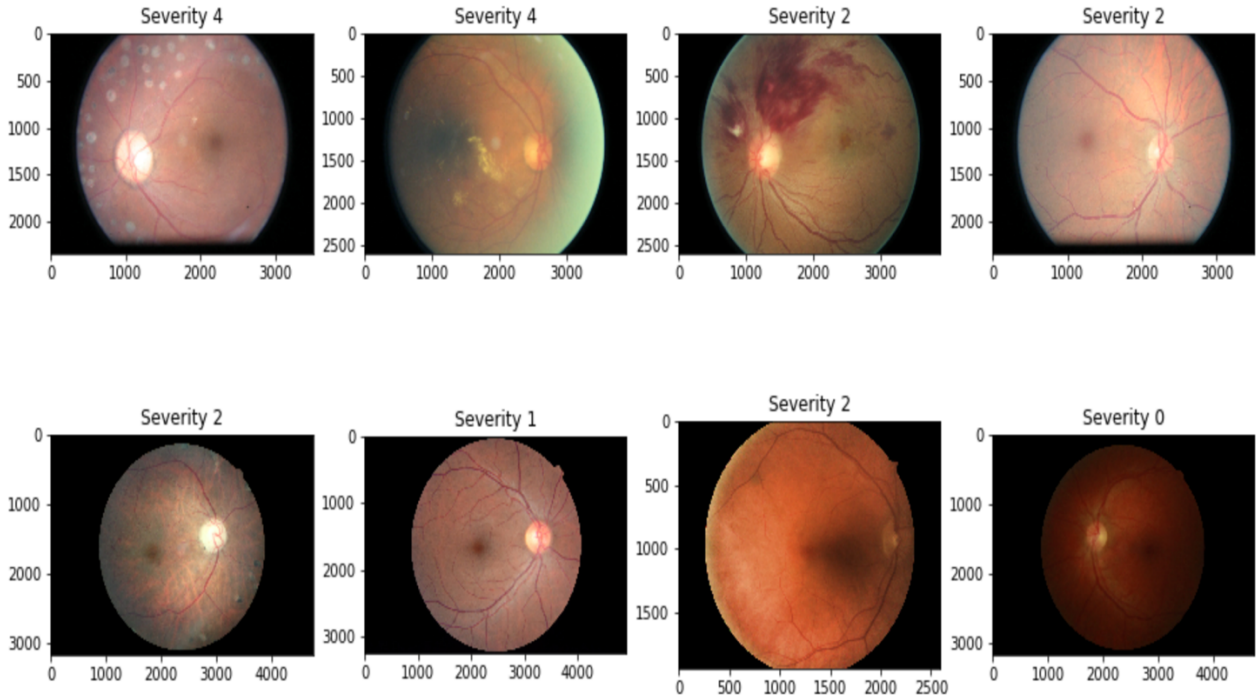


Figure 5 resized Image with the 512 X 512 resolution and Severity

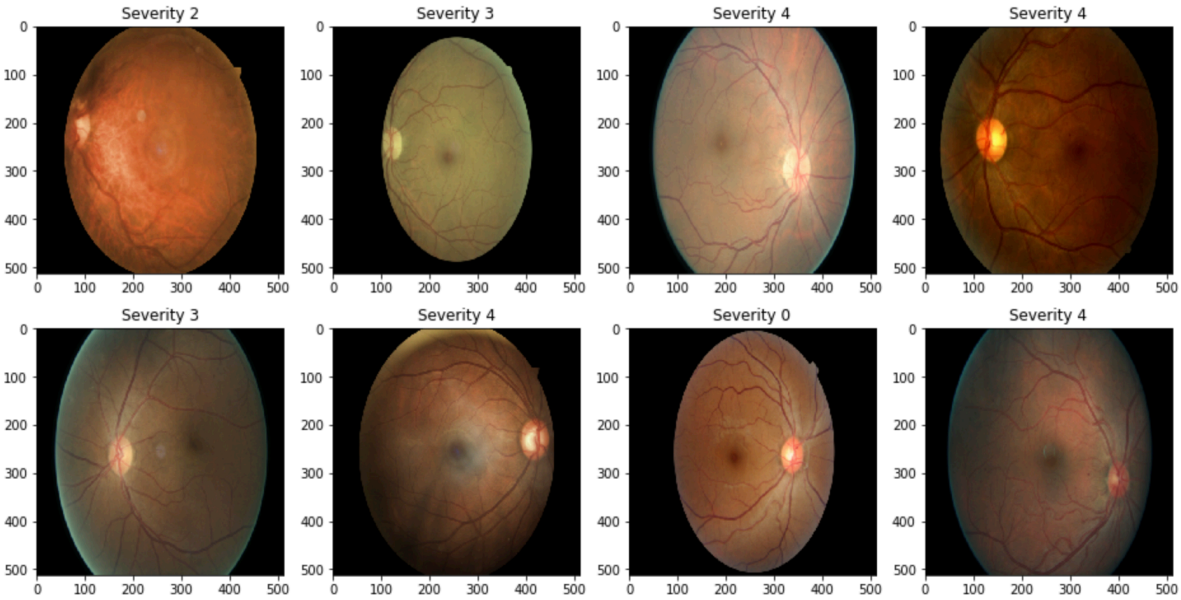
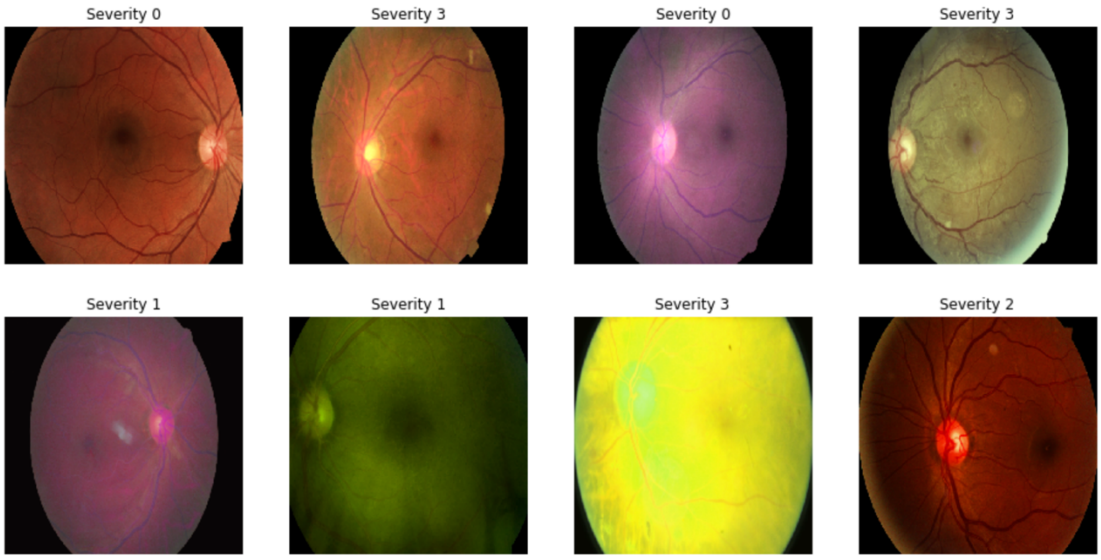


Figure 6 resized augmented Image



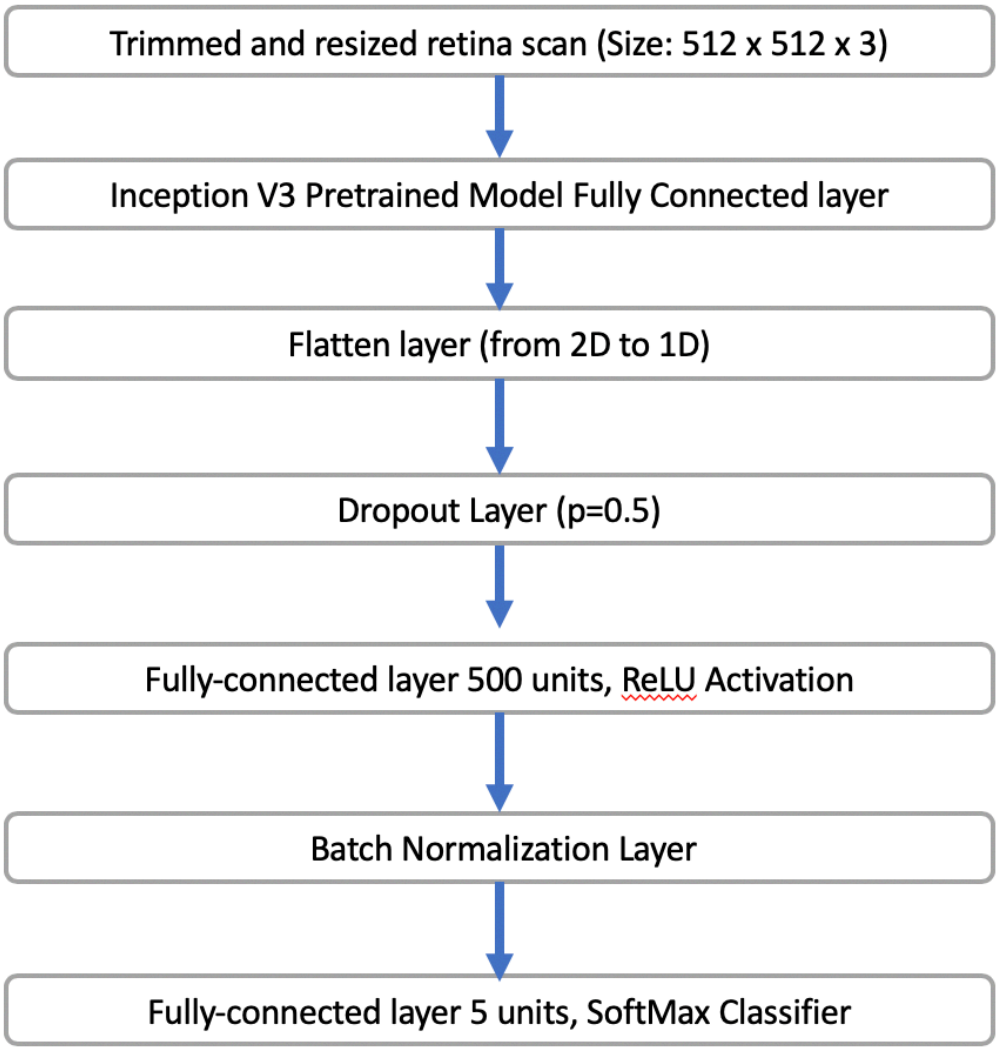
Implementation

To train the model a training dataset has been created using 75% of the original dataset. The rest of 25% has been split equally into a validation and testing dataset. The validation set is 0.25% of the total available dataset, this validation set is then shuffled and test data for prediction is picked from these validation set.

To build and train the CNN, Keras with the TensorFlow backend has been used. The InceptionV3 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.

Figure 7 The CNN Architecture of the model



To implement 'early stopping', a callback has been created which monitors the accuracy on the validation dataset. If no improvement occurs after 10 epochs the model stops training. Another callback has been created which saves the model that obtained the best accuracy on the validation dataset.

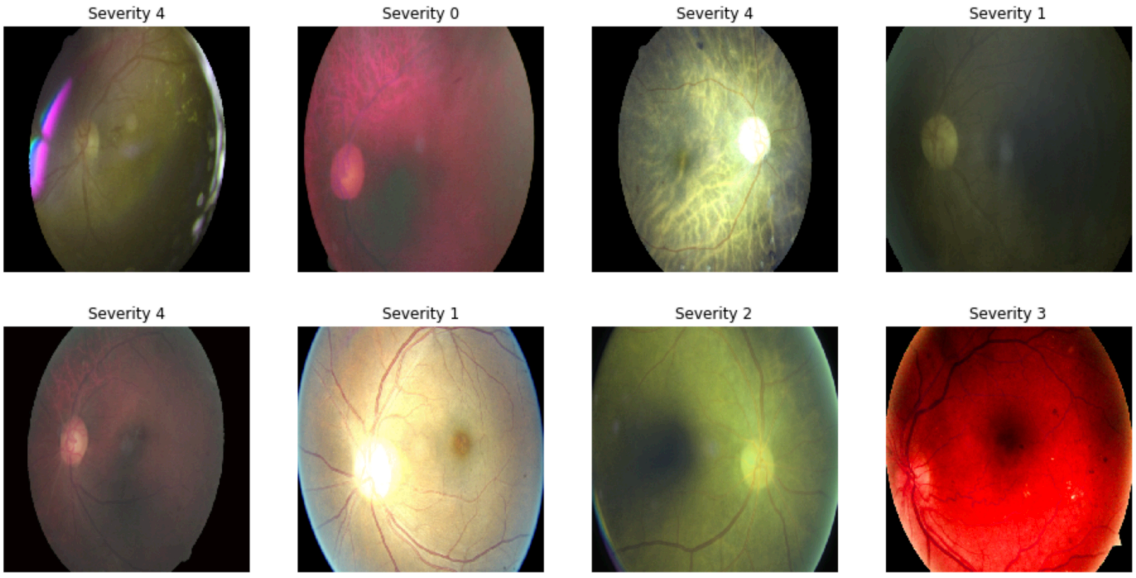
The model has been trained with learning rate of 0.01 and Adam as optimizer. To assess the performance of the model on the validation and test dataset, the categorical_crossentropy and top_k_categorical_accuracy from the keras metrics library has been used.

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

- Horizontal flip
- Vertical flip
- Rotation 5%
- Brightness and contrast
- Hue

The model has been trained using the **fit_generator** function, which generates 32 augmented images each batch, Figure 8 shows example of augmented results

Figure 8 Augmented retina images



Refinement

The refinement of the model started right with the data preprocessing step. When downscaling images to 120 x 120px, 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 512 x512px 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 25% of the reduced training data. The assessment has been done on the accuracy, specificity, sensitivity and ROC AUC curve scores in order to compare the performance of the model with the winning model of this competition.

I Evaluated the model with 5952 test data following is the classification report, I'm happy with the result as it's around 74 percent given that I only trained the data on 2200 images.

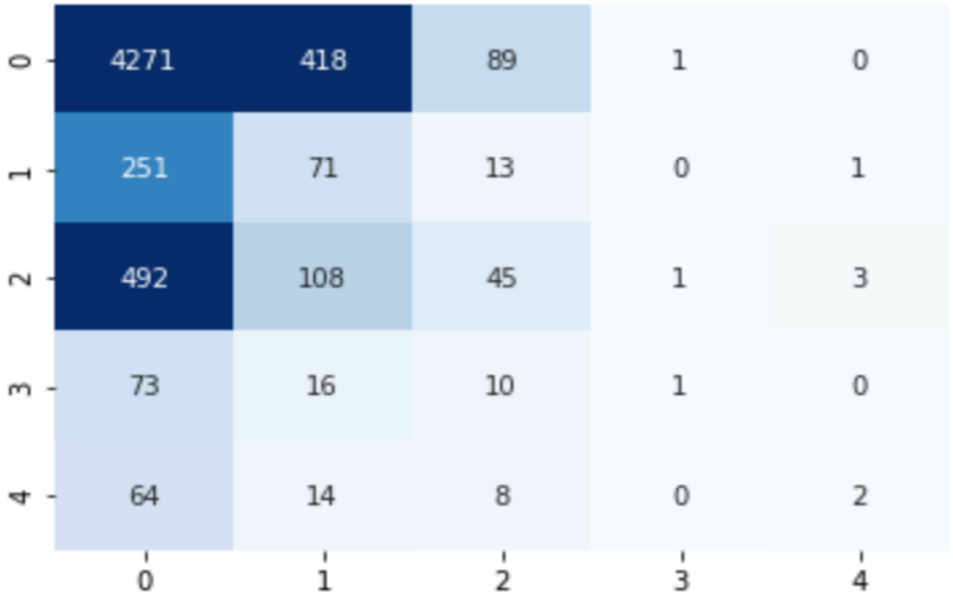
```
5952/5952 [==============================] - 182s 31ms/step
Accuracy on Test Data: 0.74%
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      4779
           1       0.11      0.21      0.15       336
           2       0.27      0.07      0.11       649
           3       0.33      0.01      0.02       100
           4       0.33      0.02      0.04        88

    accuracy                           0.74      5952
   macro avg       0.38      0.24      0.24      5952
weighted avg       0.71      0.74      0.71      5952
```

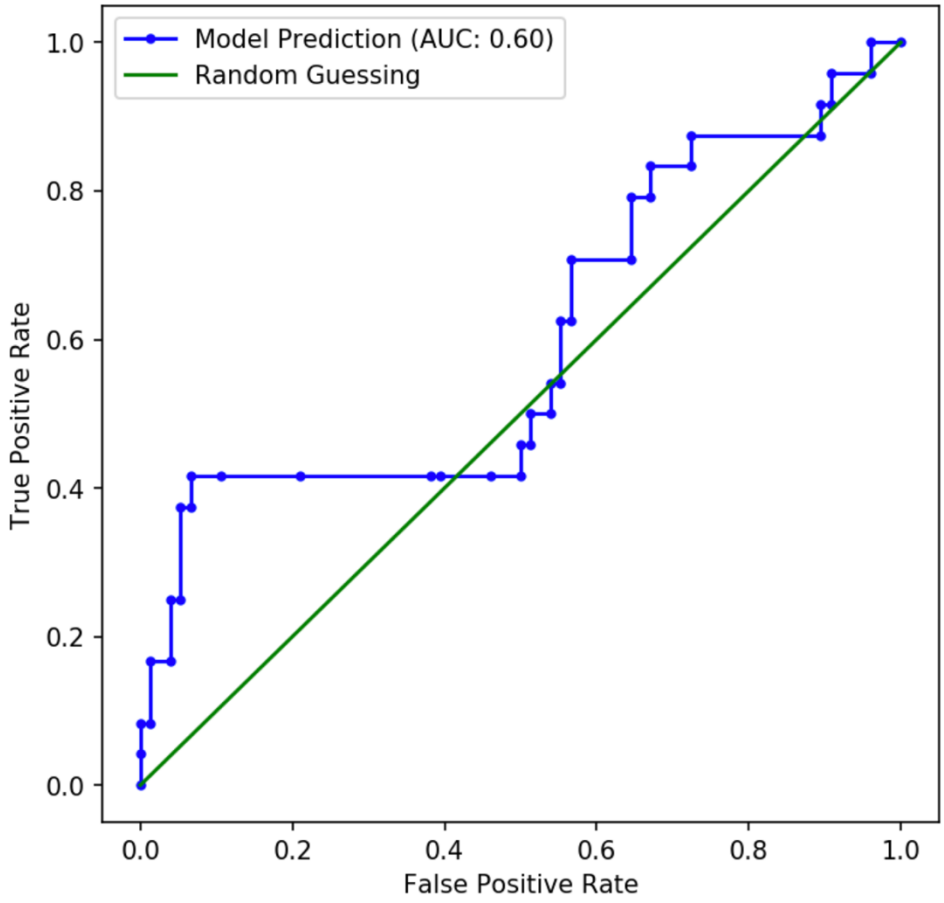
Figures 8, 9 show the confusion matrix and ROC curve obtained on above testing datasets. The confusion matrix provides a summary of how well the model assigned the samples to their ground truth label. As you can see in all three matrices, the model is confusing the classes mostly with the adjacent classes. 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.

Figure 8 Confusion Matrix of the test dataset



The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.

Figure 9: ROC curve on the test data



Sensitivity and Specificity for the above test

```
Sensitivity :  0.96
Specificity :  0.25
```

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 accuracy 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.

Justification

The winning model achieved a 0.849 accuracy 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. 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 of the model:

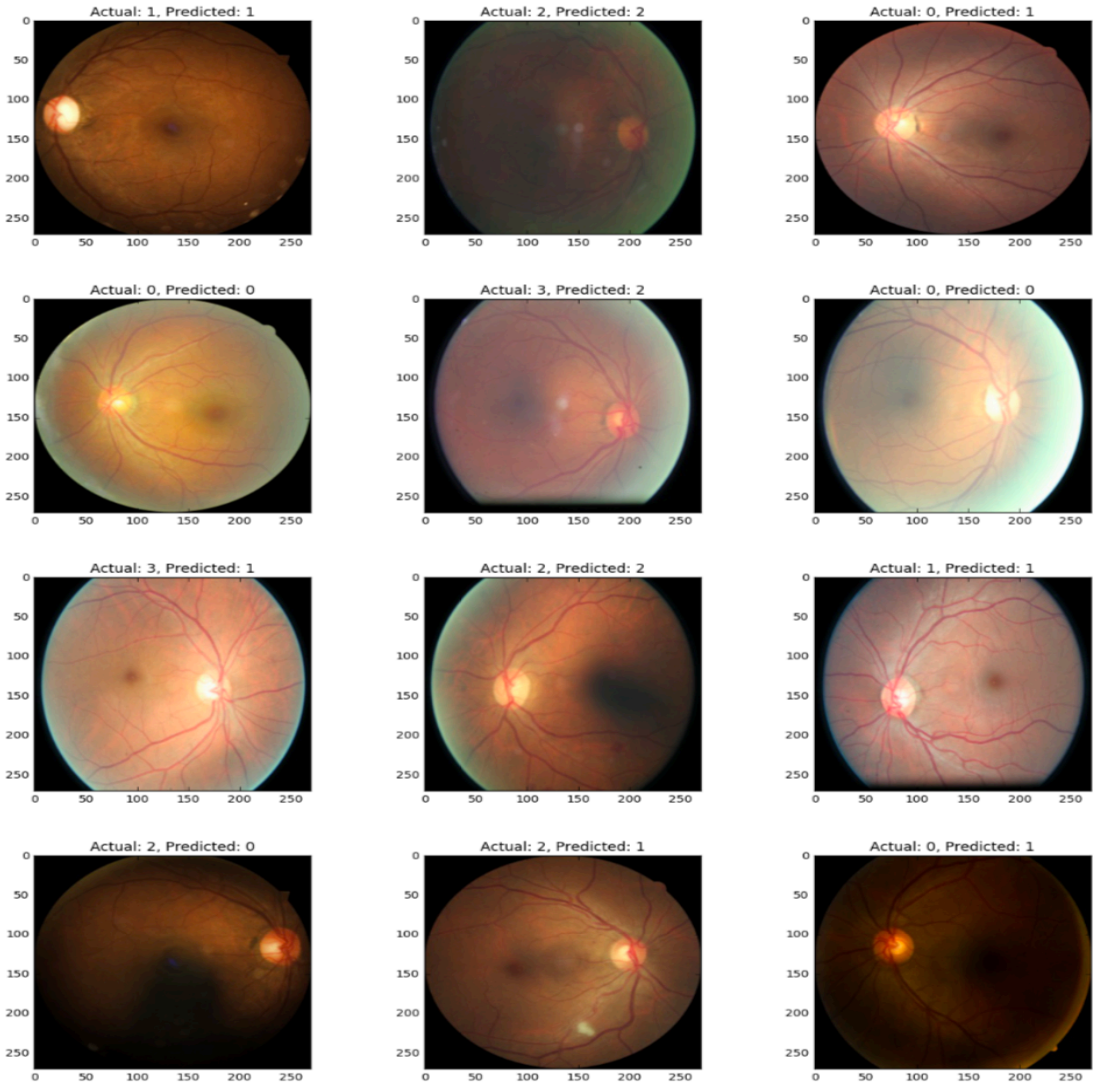
- Standardizing images with zero mean and unit standard deviation. The validation and test dataset were standardized with the mean and standard deviation of the training dataset.
- Treating the problem as a regression task rather than a classification task. Instead of using a Softmax classifier, a regression 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, ResNet-50, Xception (didn't fit into memory)
-

V. Conclusion

Free-Form Visualization

In Figure 10, 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 10 Actual vs Predicted Result



Reflection

This report presented a methodology for DR detection in retina scans using CNN.

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 long and tedious. By switching to Inception-V3 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 or Recurrent Neural Networks (RNN) with attention mechanisms

References

1] Quellec G., Lamard M., Josselin P. M., Cazuguel G., Cochener B., Roux C. Optimal wavelet transform for the detection of microaneurysms in retina photographs. IEEE Transactions on Medical Imaging, 2008;27(9):1230–1241. [PMC free article] [PubMed] [Google Scholar]

2] UR A. Decision support system for diabetic retinopathy using discrete wavelet transform. Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 2013;227(3):251–261. [PubMed] [Google Scholar]

3] Antal B., Hajdu A. An ensemble-based system for microaneurysm detection and diabetic retinopathy grading. IEEE transactions on biomedical engineering, 2012;59(6):1720–1726. [PubMed] [Google Scholar]

4] Gulshan V., Peng L., Coram M., Stumpe M. C., Wu D., Narayanaswamy A., Venugopalan S., Widner K., Madams T., Cuadros J., et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA, 2016;316(22):2402–2410. [PubMed] [Google Scholar]

5] https://www.kaggle.com/c/diabetic-retinopathy-detection/overview

6] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5961805/#r14-2840838

7] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

8] Kaggle Diabetic Retinopathy Detection Competition Report [online] Available at: https://www.kaggle.com/c/diabetic-retinopathy-detection/ dis- cussion/15801 [Accessed 20 Feb. 2017]

9] Skin cancer classification https://arxiv.org/pdf/1810.10348.pdf

10] Data Augmentation using GPU https://becominghuman.ai/data-augmentation-on-gpu-in-tensorflow-13d14ecf2b19