**MAIS 202 Winter 2020 – Deliverable 2**

In short, millions of people suffering from diabetic retinopathy, which is the leading cause of blindness among young adults. Fortunately, this can be detected with a picture of the scan of an eyeball by a doctor. However, it is very time consuming and doctors are lacking everywhere in the world. The goal of this model is the predict the severity of the disease on a scale from 0 to 4 (0 for no disease and 4 for proliferate diabetic retinopathy) in order to prevent blindness.

The data is composed of retina images. Similar to the real-world dataset, these images come in different lighting, sizes, may contain artifacts, be out of focused, underexposed or overexposed. There is a total of 3662 images from the training dataset and 1928 of test images. There is a train.csv and test.csv where first csv file contains two columns (id\_code and diagnosis) and the latter only contains the id\_code from the test images folder. The id\_code is the name of the image files from the train/test folders of images. First, to preprocess the data, I loaded the training csv file into a pandas dataframe. After loading it, I added the entire path of the images to the column id\_code. I iterated over the inputs (i.e. id\_code column) and using the path, I wrote a function which would read every image from the input *pandas* series and represent it as an array. The function returns an array of all the resized matrices representing the images in the specific dataset. I resized all the images so they can share a common size since these images all have different sizes and it would be computationally heavy since some of them are very large. I represented the images as matrices since it will be easier to work with numbers.

I created a model using pretrained VGG16. I also added some layers to the base model. Then, I trained my model using the matrices representing the images as the input and the given labels as the expected output. I also split the data using a ratio of 0.20 for the validation set and 0.80 for the training dataset. I trained my model with the training set and validated it using the validation set for a total of 30 epochs. At first the loss from each epoch would be stuck at around 1.3. It was because the learning rate was too high. At last, I set the learning rate to 0.003 and I used a gradient descent as optimizer.

The accuracy of my model is 95.04% and a loss of .1128 for the training set. For the validation set, the model had an accuracy 85.54% of and a loss of 0.4163 (see the code for the plots).

I think that further preprocessing of the images will help improve the accuracy of the model. The images have different lighting and are not all centered. Hence, maybe working with grayscale images will help the model to learn the correct weights.