Diabetic Retinopathy Detection

A Preprint

David Unger
3522491
M. Eng. Autonome Systeme
University of Stuttgart
st172353@stud.uni-stuttgart.de

Nick Wagner
3524444
B. Eng. Autonome Systeme
University of Stuttgart
st175644@stud.uni-stuttgart.de

February 12, 2022

Abstract

Diabetic retinopathy is an eye disease that can affect people suffering diabetes. It causes damage to the blood vessels of the eyes, deteriorates the eyesight and can lead in the worst case to blindness of the patient. It is important to detect the disease in an early stage to mitigate it as good as possible with an early treatment. Analyzing images of eyes and classify the severity of diabetic retinopathy is a challenging task that requires expert knowledge. To assist doctors and medical personnel, a classification model shall be trained to classify the severity automatically.

1 Introduction

Diabetic retinopathy is a complication of diabetes, which can cause damage the retina of the eye. If not detected early, this damage may cause cause vision impairment or even blindness. To treat this condition successfully, it has to be detected at an early stage, which is difficult due to minimal to no early warning signs. Furthermore, the different grades can only be distinguished by a trained professional due to its subtle symptoms. Examples are leaking blood vessels, fatty deposits or retinal swelling. Since this task is difficult even for trained professionals, an algorithm for automatic detection of the diabetic retinopathy grade is necessary. This is the goal of this paper.

The used dataset is the Indian Diabetic Retinopathy Image Dataset (IDRID), which is publicly available. It contains five class labels, which refer to the different eye disease grades (0-4).

2 Object Classification

2.1 Problem analysis

To tackle the problem of diabetic retinopathy detection, several methods are possible. Because the dataset consists of ordinally scaled data of 5 classes, regression could be used to estimate the serverity of a case. In addition, a the problem can be handled as a classification problem after one-hot-encoding the labels. As a third option, one can define a threshold to define problematic diabetic retinopathy and non-problematic diabetic retinopathy and can handle the problem as a binary classification. Further, only binary and multiclass classification are anaylzed.

A binary classification has the advantage of higher accuracy, but lacks details, because the network only outputs 0 or 1 and no information about the exact serverity of the disease. Metrics are also easy to implement, because precision, recall and f1-score are standard implementations and nicely interpretable.

A multiclass classification has typically a lower accuracy, because the network needs to pick the right class among several classes. It provides the benefit or receiving richer information, i.e. the exact serverity of the disease. Evaluating a multiclass classification problem becomes harder, because missclassifications can vary in their error. Classifiying a class 1 as class 2 is for example less problematic than classifying class 1 as class 5.

2.2 Architecture

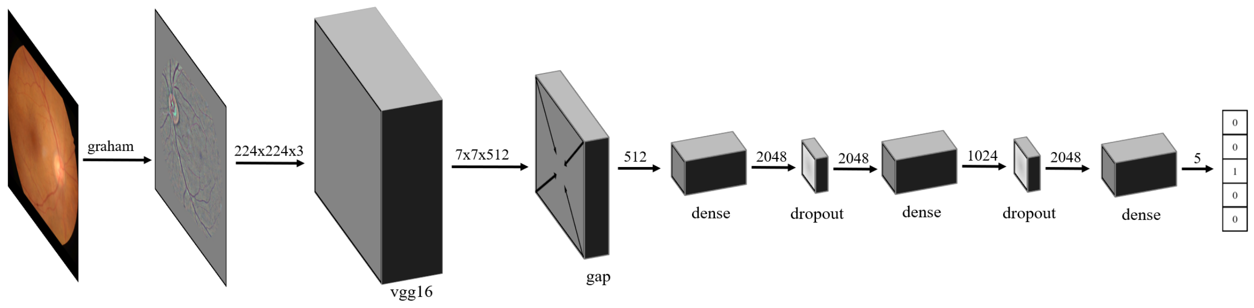


Figure 1: Model architecture

The center piece of the architecture is a vgg16 transfer model that is trained on imagenet data. It's responsibility is to learn local features before the feature maps get average pooled. Then three dense layers including dropout for regularization are used. The final output consists of five values that correspond to the five diabetic retinopathy grade predictions. The decision for this architecture is based on parameter optimization of many factors which get discussed in chapter 3.2. » draw 3D architecture in powerpoint

To tackle the classification problem, we analyzed several model architectures that are composed of three parts: the base model, the conversion layer and the head of our model. For the base model we used ResNet50 and VGG16 with weights pretrained on ImageNet. We tried out the results with a fully trainable, partly trainable and not trainable base model. As conversion layer we tested both, a flatten layer and global average pooling (GAP). For the head we tried up to three hidden dense layers with different number of neurons per layer.

2.3 Weight initialization

Weight initialization refers to the initial values of parameters that are used in specific neural network layers. Changing the initialization of the layers changes the starting point for the optimization process and potentially also the performance. Keras initializes the weights of dense layers with the Glorot uniform initializer, which draws samples from a truncated uniform distribution. Generally the goal is to avoid vanishing and exploding gradients, even better is if the variance of layer outputs is approximately one. [paper KUMAR] Specifically for the ReLU activation function, this is achieved with the He initialization, which draws from a normal distribution with the following parameters. [paper HE]

$$\mu = 0 \qquad (1) \qquad\qquad \sigma^2 = 2/N \qquad (2)$$

This led to a faster training and performance increase of X percent.

2.4 Augmentation

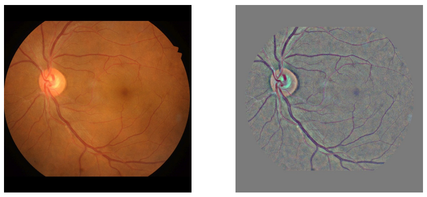
Within the input pipeline three main types of augmentation are applied. The goal is to make feature extraction easier for the network and increase the amount of input images which reduces overfitting.

- **Graham preprocessing: [report kaggle]**
  1. rescale the eye radius to 300 pixels
  2. subtract the local average color such that the local average gets mapped to gray
  3. clip the image to remove boundary effects]
- **Color jittering:** Slight random changes to brightness, hue, saturation or contrast.
- **Random cropping:** Crop a window that is slightly smaller than the image. Then resize to original size again.

2.5 Dataset Balancing

Taking a closer look at the sample distribution within the dataset 1, it is obviously imbalanced. This will inevitably lead to a trained model that is fitted better to the overrepresented classes.

To avoid this, a method called oversampling is applied. Oversampling allows drawing underrepresented classes from the training set more often, such that all classes are equally represented when training the model.



(a) Original image (b) Processed image

Figure 2: Graham pre-processing

Label:	0	1	2	3	4
# train	134	20	136	74	49
# test	34	5	32	19	13

Table 1: Dataset sample distribution

2.6 Training

For training our model, we used categorical cross-entropy as loss function and experimented with the optimizer and learning rate. As optimizer we selected SGD with a momentum of 0.9 and Adam which both performed quite similar.

As learning rate, we tried 0.1, 0.01 and 0.001 from which latter performed best and further learning rate decay showed no further improvement.

2.7 Metrics

Training deep neural networks requires some performance metrics indicating the success or failure of the model fitting. Accuracy is easy to understand and to implement, but not suited for imbalanced data, because it doesn't take into account how well each single class is predicted. To overcome this, precision (how accurate was the model with its prediction) and recall (how thorough is a model with its prediction) can be used. Especially a combination of both, the f1-score is a metrics that indicates how well a classifier handles different classes.

For the diabetic retinopathy dataset, the f1-score comes with a drawback. The dataset is ordinally scaled, which means predicting a wrong class is not equally bad. As an example is predicting class 4 for the true class 0 much worse that predicting a 1. The f1-score would count both missclassifications equally, while the Quadratic Weighted Kappa (QWC) metric takes the distance of the classification error into account.

3 Experiments

3.1 Procedure

The training of the deep neural network classifier requires the selection of suitable hyperparameters that differ from problem to problem. A useful strategy to find a good set of hyperparameters are parameter sweeps. Weights&Biases is a python library that enables the easy implementation of sweeps.

Hyperparameter optimization requires besides training and test dataset a third, the validation dataset, to evaluate the model after hyperparameter tuning and to avoid overfitting on the hyperparameters. Because the given dataset only contains training and test data, the original training dataset was split into 80% training data and 20% validation data.

To perform many parameter sweeps, we stored out dataset in the cloud and used Google Colab as training server.

3.2 Hyperparameter selection

As metrics for hyperparameter selection, we mainly used correlation with the validation-f1-score and compared the minimum, median and maximum run of each sweeped hyperparameter value with each other. Once most important hyperparameter is determined and selected, one needs to ensure that the second most important parameter is not selected because of strong correlation with the most important one. Therefore, the independence can be guaranteed by filtering the sweep and selecting just runs with the best parameter one or by performing a new parameter sweep with a fixed parameter one.

With this procedure, we selected the most important parameters as in table...:

Hyperparameter	Selected
Use Graham prepreccesing	False
Model architecture	ResNet50
Learning rate	0.001
Balancing	true
Flatten or GAP	GAP
Number of dense neurons in layer N-3	4096
Number of dense neurons in layer N-1	0

Table 2: Dataset sample distribution

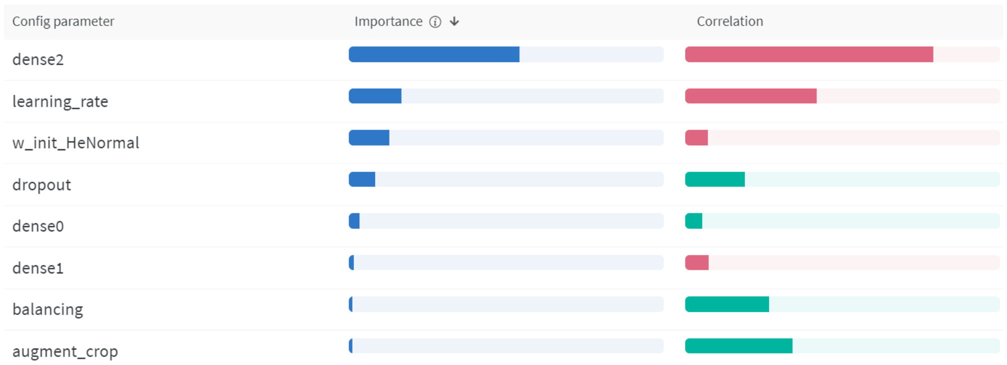


Figure 3: Most important parameters with a sweep of 80 experiments

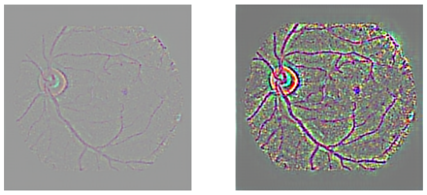
The following parameters show a big effect on the performance of the neural network on the validation data, why they are selected for the final classifier.

Balancing ...

3.3 Deep Visualization

3.3.1 Guided Backpropagation

Guided Backpropagation belongs to the family of pixel-space gradient visualizations. The goal is to exploit the idea that neurons act like detectors of image features by using backpropagation. What makes this backpropagation guided is that negative gradient are set to zero. This way, only pixels that are positively important to the output get highlighted. This method is not class-discriminative.



(a) Original result (b) Enhanced result

Figure 4: Guided Backpropagation

3.3.2 CAM

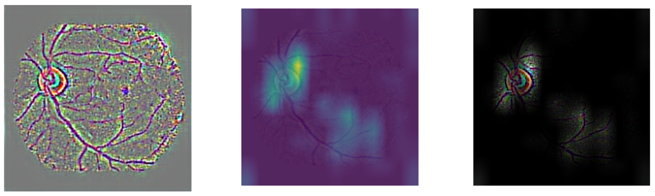
Class Activation Map (CAM) is class-discriminative and highlights relevant image regions, but in a less fine-grained manner To achieve this, global average pooling has to be performed on the last feature maps of the last convolutional layer, followed by a dense layer as output. The weights of this dense layer are then projected back to the convolutional feature maps which results in the class activation mappings. As a consequence, it is hard to generalize this approach due to the architecture restrictions.

3.3.3 GradCAM

GradCAM is class-discriminative as well, but can be used by any CNN-based network without architectural adaptions. Similar to CAM, it describes the activation as a linear combination of weighted feature maps. But in this case, the weights are calculated by deriving the logit per class by the feature maps of the chosen convolutional layer. These gradients then get global average pooled, followed by a ReLU. The ReLU leaves only feature with a positive impact behind, similar to how it is applied in Guided Backpropagation.

3.3.4 Guided GradCAM

Guided GradCAM combines the ideas of Guided Backpropagation with Class Activation Maps. This makes GradCAM class-discriminative due to the localization of relevant image regions and high-resolution due pixel-space gradient visualization.



(a) Guided Backprop. (b) gradCAM (c) Guided gradCAM

Figure 5: Guided Backpropagation

Note: The chosen model architecture leads to graph issues within the gradCAM algorithm. The workaround affects the result negatively. A working example with another model and image can be found in the code within the "helper" folder.

4 Results

We trained and evaluated our algorithm on both, binary and multiclass classification.

For binary classification we performed a hyperparameter sweep and achieved without further optimization an f1-score of 0.87 on the validation data and of 0.82 on the test data.

		Predicted				
		Class 0	Class 1	Class 2	Class 3	Class 4
Actual	Class 0	29	2	2	0	1
	Class 1	0	3	2	0	0
	Class 2	7	1	15	3	6
	Class 3	2	0	3	9	4
	Class 4	4	0	1	1	7

For multiclass classification we optimized our model further and performed several sweep. Our finally selected model achieves a QWC score of 0.44 which would be rank 106 for the Kaggle challenge. best binary + multiclass performance; color coded confusion matrix

References

[1] Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv:1704.08863*, 2017.

[2] name. title. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.
Guided Backpropagation: https://arxiv.org/abs/1412.6806