DIABETIC RETINOPATHY DETECTION

A PREPRINT

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

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

February 13, 2022

ABSTRACT

Diabetic retinopathy is an eye disease that can affect people suffering from diabetes. To assist doctors and medical personnel with their diagnosis, a classification model shall be trained to classify the severity of the disease without human interaction. We train and analyze several convolutional neural networks with different architectures. To find an optimal architecture, intensive hyperparamer sweeps via random search are performed. To analyze the performance of our model, we use gradient visualization.

1 Introduction

Diabetic retinopathy is a complication of diabetes, which can cause damage to the retina of the eye. To treat this condition successfully, it has to be detected at an early stage, which is difficult due to minimal or no early warning signs. Furthermore, the different grades can only be distinguished by a trained professional due to its subtle symptoms. Examples for those are leaking blood vessels, fatty deposits or retinal swelling. Since this task is difficult even for trained professionals, the goal of this paper is to find an algorithm for automatic detection of the diabetic retinopathy grade. The dataset is the Indian Diabetic Retinopathy Image Dataset (IDRID), which is publicly available[QUELLE]. It contains five class labels, which refer to the each of the 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 severity of a case. In addition, the problem can be handled as a binary or multiclass classification problem after one-hot-encoding the labels. Further, only binary and multiclass classification are analyzed.

A binary classification has the advantage of higher accuracy, but lacks details, because the network doesn't output information about the exact severity of the disease. Metrics are 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 of receiving richer information, i.e. the exact severity of the disease. Evaluating a multiclass classification problem becomes harder, because misclassifications can vary in their error.

2.2 Architecture

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

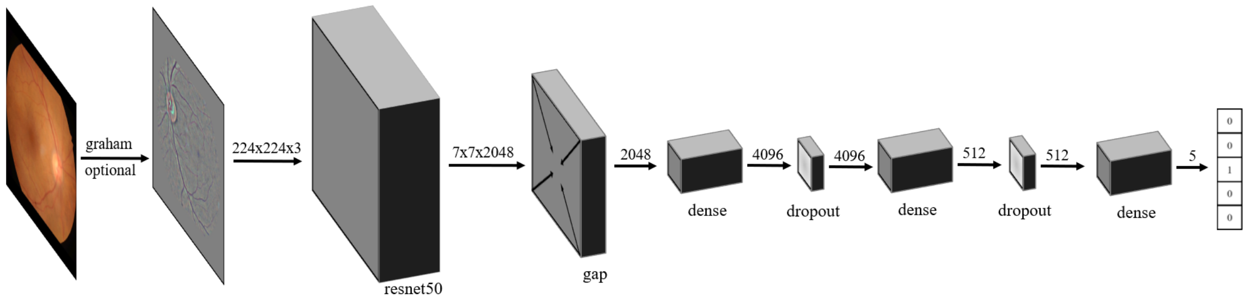


Figure 1: Model architecture

2.3 Weight initialization

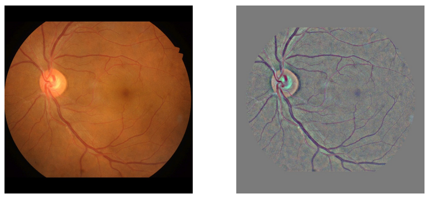
Weight initialization refers to the initial values of parameters that are used in neural network layers. Changing the initialization of the layers changes the starting point for the optimization process and potentially also the performance. Keras initializes the weights of dense layers with the Glorot 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. [QUELLE KUMAR] Specifically for the ReLU activation function, this is achieved with the He initialization, which draws from a normal distribution with the following parameters. [QUELLE paper HE]

$$\mu = 0 \tag{1}$$
$$\sigma^2 = 2/N \tag{2}$$

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



(a) Original image (b) Processed image

Figure 2: Graham pre-processing

2.5 Dataset Balancing

Taking a closer look at the sample distribution within the dataset as shown in Table 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.

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 & Metrics

For training our model, we use categorical cross-entropy as loss function and experiment with the optimizer and learning rate. As optimizer we select SGD with a momentum of 0.9 and Adam which both show similar performance. As learning rate, we use 0.1, 0.01 and 0.001 from which the latter performs best. Further learning rate decay shows no improvement.

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 the model is with its prediction) and recall (how thorough the model is with its prediction) can be used. Especially a combination of both, the F1-score is a metric 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, predicting class 0 (no retinopathy) for the true class 4 (proliferative retinopathy) is much worse than predicting class 3 (servere retinopathy). The F1-score counts both missclassifications equally, while the Quadratic Weighted Kappa (QWC) [QUELLE] metric takes the distance of the classification error into account and is therefore more suitable.

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, which is why their choice is a hard task. 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 and is therefore selected for our implementation.

Hyperparameter optimization requires besides the 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 is split into 80% training data and 20% validation data. To perform many parameter sweeps, we store our dataset in the cloud and use Google Colab as training server.

3.2 Hyperparameter selection

As metrics for hyperparameter selection, we mainly use correlation with the validation-QWC-score and compare 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. This independence can be guaranteed by filtering the sweep and selecting just runs with the best parameter or by performing a new parameter sweep with having this first parameter fixed.

With this procedure, we selected the most important parameters as in Table 2.

Hyperparameter	Selected
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-2	512
Number of dense neurons in layer N-1	0

Table 2: Excerpt of the most important hyperparameters

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 gradients are set to zero. This way, only pixels that are positively impacting the output get highlighted. This method is not class-discriminative. [QUELLE]

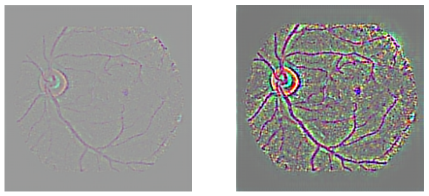

(a) Original result (b) Enhanced result

Figure 3: 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 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 adaptations. 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 respective logit of the class of interest 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.[QUELLE]

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. [QUELLE]

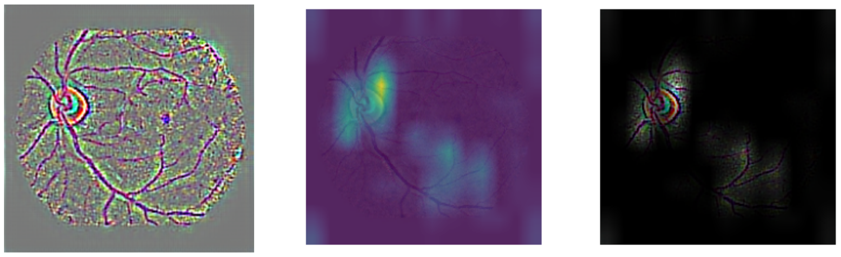

(a) Guided Backprop. (b) gradCAM (c) Guided gradCAM

Figure 4: 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 train and evaluate our algorithm on both binary and multiclass classification.

For binary classification we perform a hyperparameter sweep and achieve without further optimization an F1-score of 0.87 on the validation data and of 0.82 on the test data.

For multiclass classification we optimize our model further and perform several sweeps. Our finally selected model achieves an accuracy of 58.8%, an F1-score of 0.47 and a QWC score of 0.44 which would have achieved rank 106 for the Kaggle challenge. When evaluating the confusion matrix of the prediction as denoted in Table 3, one can see that our model predicts class 0 very frequently and often incorrectly. This can be interpreted in a way that our model fails to identify features that indicate the disease and therefore selects class 0. As a result, our model could be further optimized by more extensive data augmentation like shearing or rotating the images to overcome the challenge of a relatively small dataset.

		Predicted				
		Class 0	Class 1	Class 2	Class 3	Class 4
True	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

Table 3: Confusion matrix on the test dataset

References

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

[2] Porwal, Prasanna; Pachade, Samiksha; Kamble, Ravi; Kokare, Manesh; Deshmukh, Girish; Sahasrabuddhe, Vivek; Meriaudeau, Fabrice. Indian diabetic retinopathy image dataset (IDRiD): a database for diabetic retinopathy screening research. *Data 3*, 2018.