Udacity ML Engineer Nanodegree

Capstone Project

Detecting Blindness with Deep Learning

Nikita Kozodoi

May 2020

Contents

1 Definition

1.1 Project Overview

Diabetic retinopathy (DR) is one of the leading causes of vision loss. According to a recent study from International Diabetes Federation, the global prevalence of DR among the individuals with diabetes for the period from 2015 to 2019 was at more than 25% [5]. According to the World Health Organization, more than 300 million people worldwide have diabetes, and the disease prevalence has been rising rapidly in developing countries [6].

Early detection and treatment are crucial steps towards preventing DR [2]. Currently, detecting DR is a time-consuming process. The screening procedure requires a trained clinical expert to examine the fundus photographs of the patient's retina. This creates delays in diagnosis and treatment of the disease. Automated evaluation of retina photographs can speed up the efficiency and coverage of the DR screening programs. This is especially relevant for developing countries, which often lack qualified medical stuff to perform the diagnosis.

The project aims at developing a deep learning model for predicting the severity of DR disease based on the patient's retina photograph. Previous research has explored the usage of deep learning for detecting DR and concluded that convolutional neural networks (CNNs) have high potential in this task [2]. The Asia Pacific Tele-Ophthalmology Society (APTOS) has launched two Kaggle competitions with a goal of promoting the use of deep learning for DR detection and boosting the development of automated detection systems.

The project leverages two data sets: (i) main data set used for modeling and evaluation; (ii) supplementary data set for pre-training. The main data set is provided by APTOS. It has been employed in the APTOS 2019 Blindness Detection competition on Kaggle and is available for the download at the competition website: `https://www.kaggle.com/c/aptos2019-blindness-detection/data`. The data set includes 3,662 labeled retina images of clinical patients. The images are taken using a fundus photography technique.

The images are labeled by a clinical expert. The labels indicate the severity of DR from 0 to 4 in accordance with the scale provided in Section 1.2. The data also include a test set with 1,928 images whose labels are not explicitly disclosed. The test set can be used for performance evaluation using a «late submission» functionality at the Kaggle competition website.

The supplementary data set features 35,126 retina images labeled by a clinician using the same scale as the main data set. The data set has been used in the 2015 Diabetic Retinopathy Detection competition and is available for the download at the corresponding website: `https://www.kaggle.com/c/diabetic-retinopathy-detection/data`.

1.2 Problem Statement

The project aims at developing a deep learning model for predicting the severity of the DR disease based on the patient's retina photograph. Severity of DR determines the necessary actions for prevention and/or required treatment. Therefore, we consider the multiple classification task, in which five disease stages are distinguished based on its severity:

- 0 – no disease
- 1 – mild stage
- 2 – moderate stage
- 3 – severe stage
- 4 – proliferative stage

1.3 Metrics

The project considers DR detection as an ordinal classification task. In such a setting, a suitable evaluation metric is the Cohen's Kappa, which measures the agreement between the actual and predicted labels [1]. The metric varies from 0 (random agreement) to 1 (perfect agreement).

The Kappa is computed as follows. First, one constructs three matrices: (i) the matrix of observed scores X; (ii) the matrix of expected scores based on chance agreement M; (iii) the weight matrix W. Next, the Kappa is computed as:

$$\kappa_w = 1 - \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} x_{ij}}{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} m_{ij}}, \tag{1}$$

where x_{ij} and m_{ij} are elements in the observed and predicted matrices, respectively, whereas the w_{ij} are weights. wewill use Kappa with quadratic weights to stronger penalize larger errors.

2 Analysis

2.1 Data Exploration

The data set is collected from multiple clinics in India using a variety of different camera models, which creates discrepancies in the image resolution, aspect ratio and other parameters. This is demonstrated in Figure 1 below, where we plot the histograms of image width, height and aspect ratio. A high variety in the image parameters requires to be accounted for during the data prepossessing. All retina images are colored images with three RGB channels.

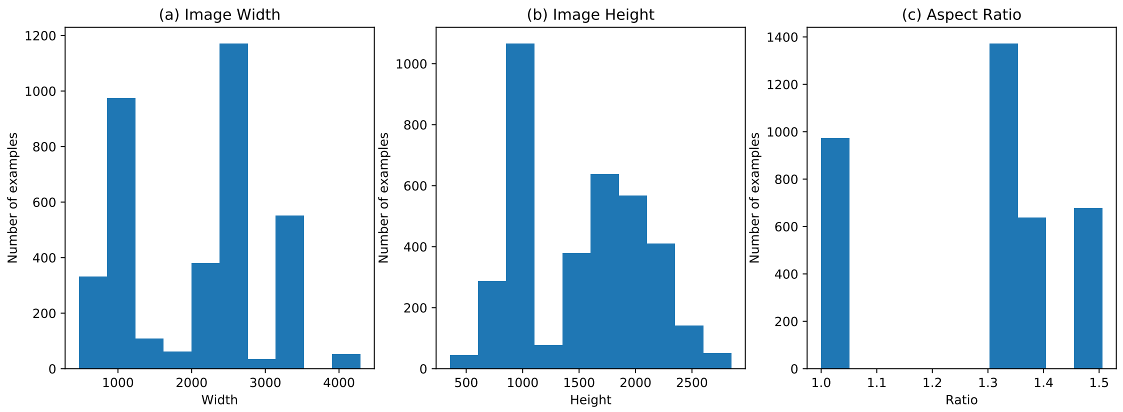

Figure 1. Image Size Distribution

The data set is imbalanced. Figure 2 illustrates the class proportions in the labeled data. 49% of images are healthy patients. The remaining 51% are pictures with different stages of DR. The least common class is 3 (severe stage) with only 5% of the total examples. The overall sample size of the labeled data set is 3,662 images.

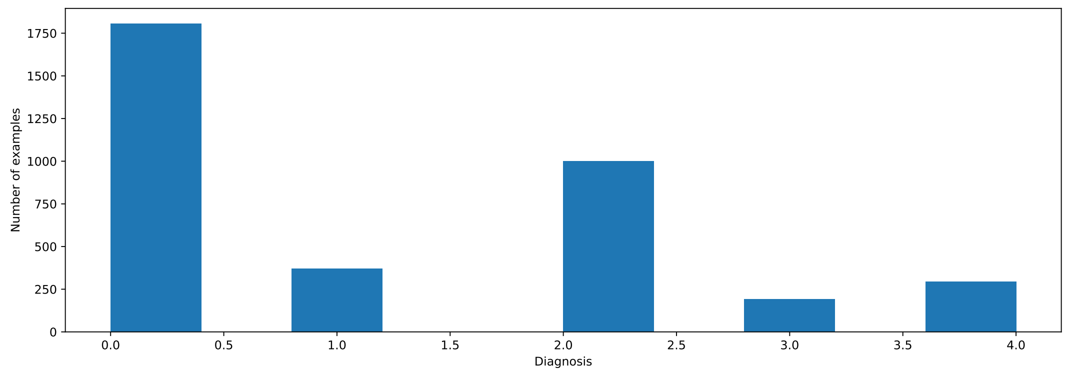

Figure 2. Class Distribution

2.2 Exploratory Visualization

In practice, the severity of DR is diagnosed by the presence of different visual cues on the retina photographs. This includes signs like abnormal blood vessels, hard exudates and so-called «cotton wool» spots. Figure 3 illustrates the exemplary retina photograph with present DR signs used to guide doctors.

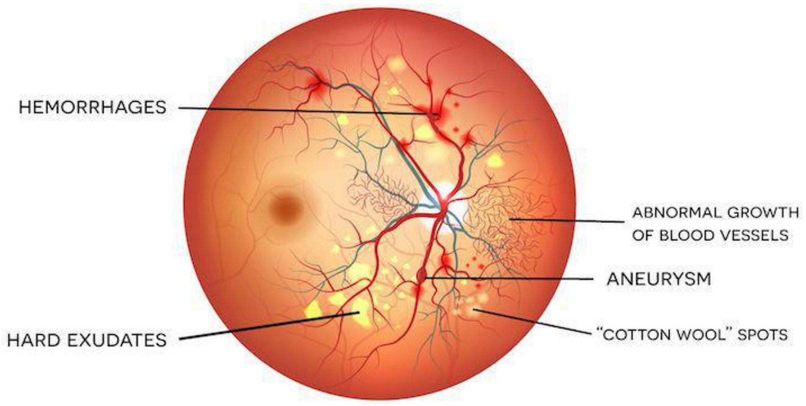

Figure 3. Illustration of the Visual Signs of DR. Source: `https://www.eyeops.com/contents/our-services/eye-diseases/diabetic-retinopathy`

Figure 4 visualizes a batch of sample images from the main data set. The labels are provided in the image titles. Comparing images of different classes, we can see the presence of exudates and «cotton wool» spots on some of the retina photographs of sick patients.

The illustration further emphasizes the difference between the retina photographs in terms of aspect ratio, lighting conditions and camera quality. We will discuss ways to tackle these discrepancies in Section 3.1.

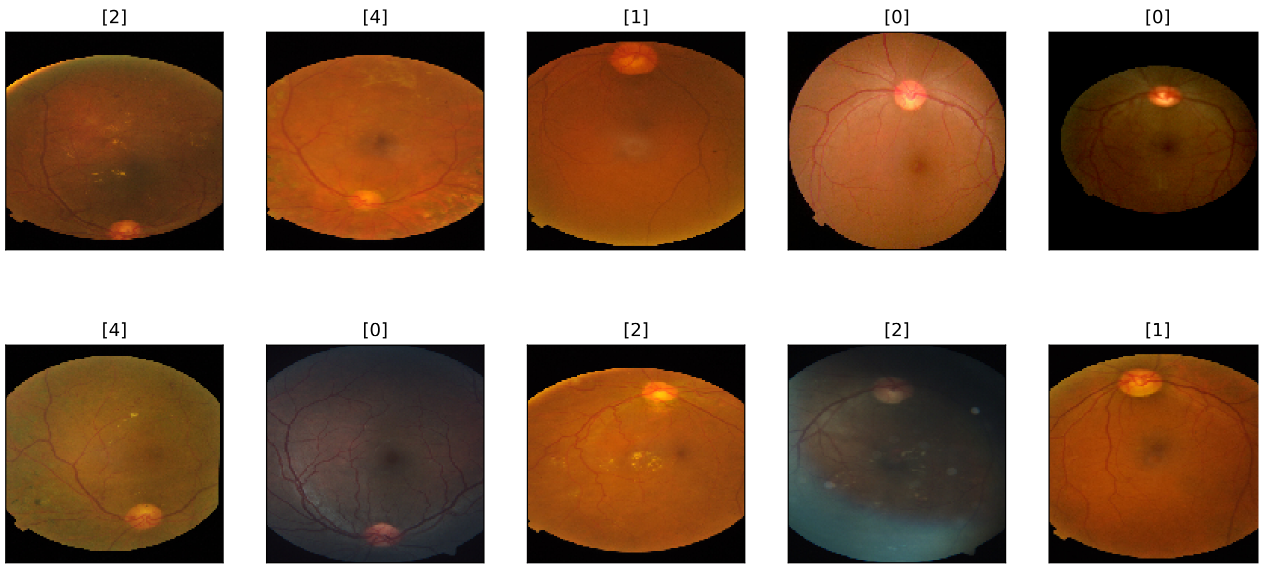

Figure 4. Data Illustration

2.3 Algorithms and Techniques

This project develops a CNN-based deep learning model for the DR severity classification task. CNNs achieve state-of-the-art performance in computer vision tasks such as image classification. Recent medical research also shows a high potential of applying CNNs in the DR classification task [2]. Furthermore, using CNNs allows to take advantage of transfer learning techniques, which is relevant given a small sample size of the considered data set.

The project employs the EfficientNet architecture for the CNN classifier. EfficientNet is one of the recent state-of-the-art CNN models in image classification [4]. EfficientNet encompasses 8 architecture variants (B0 to B7) which differ in the model complexity and default image size.

The architecture of EfficientNet B0 is presented in Figure 5. We test multiple EfficientNet neural network architectures and use the one that demonstrates the best performance.

The modeling pipeline consists of the three main stages outlined below:

1. **Pre-training the model on the 2015 data set.** The main 2019 data set has a limited number of images (N = 3,662). Therefore, we pre-train the CNN model on a larger data set from the 2015 competition (N = 35,126).

2. **Fine-tuning the model on the 2019 data set.** We fine-tune the pre-trained model on the main data set. We consider different variants of image augmentations to improve the model performance. We use cross-validation for data partitioning and make modeling decisions based on the performance observed on the out-of-fold predictions.

3. **Predicting labels of test images.** On the inference stage, we use saved weights from the base models trained on different combinations of training folds. We aggregate predictions from the base models and use test-time augmentation to further improve the predictive performance.

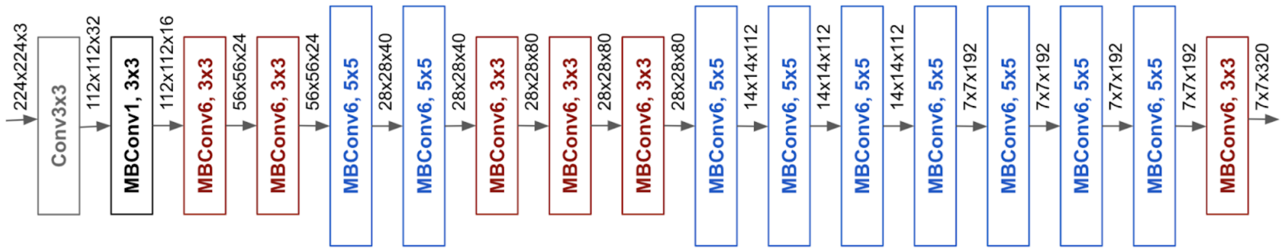

Figure 5. EfficientNet B0 Architecture. Source: `https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html`

2.4 Benchmarks

The project uses multiple benchmark models to evaluate the effectiveness of the proposed solution. We compare performance of the developed solution with the publicly available solutions derived by different teams in the APTOS 2019 Blindness Detection competition. We evaluate the model performance on the same test sample to enable direct comparison.

The particular benchmarks include:

- Fine-tuned Resnet-50 model [3] ($\kappa_w = 0.8747$)
- Optimized Resnet-50 model with train and test-time image augmentations ($\kappa_w = 0.8832$)
- EfficientNet B5 model [4] with advanced image preprocessing ($\kappa_w = 0.9098$)

3 Methodology

3.1 Data Preprocessing

As illustrated in Section 2, the retina photographs are characterized by a high variety of image parameters. This can have a strong impact on the performance of the classification model. We make multiple steps towards standardizing the images.

Visual inspection of retina images suggests that images taken from cameras with different aspect ratios result in some images having large black areas around the eye. The black areas do not contain information relevant for prediction and can be cropped. However, we see that the size of black areas varies from one image to another. To address this, we develop a cropping function that converts the image to grayscale and marks black areas based on the pixel intensity. Next, we find a mask of the image by selecting rows and columns in which all pixels exceed the intensity threshold. This helps to remove vertical or horizontal rectangles filled with black similar to the ones observed in the upper-right image in Figure 4. After removing the black stripes, we resize the images to the same height and width.

Another issue is the eye shape. Depending on the image parameters, some eyes appear to have a circular form, whereas others look like ovals. As the size and shape of items located in the retina determine the disease severity, it is crucial to standardize the eye shape as well. To do so, we develop another cropping function that makes a circular crop of a particular radius around the center of the image.

Finally, we correct for the lightning and brightness discrepancies by smoothing the images using a Gaussian filter from the cv2 image preprocessing library.

Figure 6 depicts the example batch of eye images after preprocessing. Comparing the images to the ones presented in Figure 5, we can see that the apparent discrepancies between the photographs are now fixed.

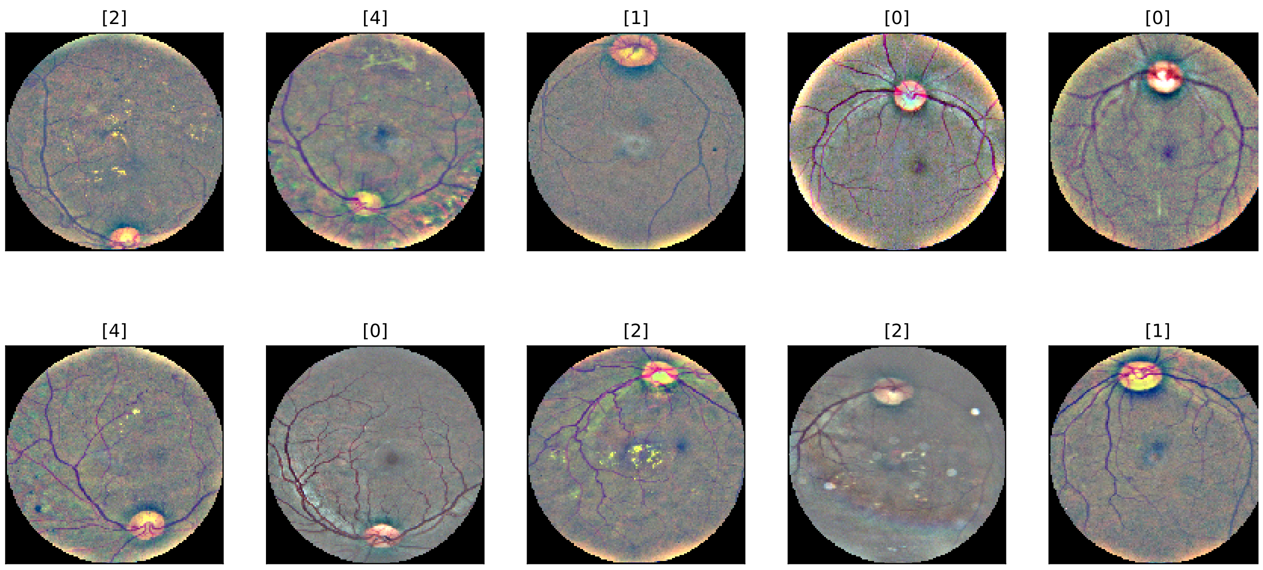

Figure 6. Data Illustration After Preprocessing

To reduce overfitting and improve the generalization ability of the model, we apply data augmentations during model training. We use the following augmentations:

- Random rotation of he image by up to 360 degrees
- Random horizontal flip of the image with a probability of 0.5
- Random vertical flip of the image with a probability of 0.5

3.2 Implementation

The models are implemented in Python 3.7 using PyTorch library. All implementations are provided in Jupyter notebooks in the «codes» folder. This section discusses the implementation process of pre-training, fine-tuning and inference stages of the modeling pipeline.

3.2.1 Pre-training

The first stage is pre-training a CNN model on the supplementary 2015 data set. First, we instantiate the EfficentNet B4 model. We initialize the model weights using the parameters pre-trained on ImageNet by downloading the model architecture and weights in the PyTorch format. The convolutional part of the network responsible for feature extraction outputs a tensor with 1792 features. To adapt the trained EfficentNet to our task, we replace the last fully-connected classification layer with a (1792, 5) fully-connected layer.

Since we are dealing with a multiple classification problem, we use cross-entropy as a loss function. We implement nn.CrossEntropyLoss() function which combines logsoftmax and negative log-likelihood loss and applies them to the output of the last network layer.

Next, we define optimizer for the network. We use adaptive learning rate optimization algorithm (Adam optimizer) with a starting learning rate of 0.001. During training, we use a learning rate scheduler, which multiplies the learning rate by 0.5 after every 5 epochs. This helps to make smaller changes to the network weights when we are getting closer to the optimum. Weights of all network layers are optimized during the pre-training.

We create a dataloader object using a batch size of 20 and the image size of 256. The choice of these parameters is a trade-off between a predictive performance and resource capacity. During training, we iteratively load a batch of training examples from the supplementary data set and conduct forward and backward passes with the network. All data from the supplementary data set is used as the training sample.

After each training epoch, we validate the model on the validation examples. We extract class scores from the last fully-connected layer and predict the image class corresponding to the highest score. We use images from the main 2019 data set as a validation sample. This allows us to validate the network on the images from the main data set. We train the network for 15 epochs, tracking the validation loss and Cohen's kappa. If the kappa does not increase for 5 consecutive epochs, we terminate the training process and save model weights for the epoch which is associated with the highest validation kappa.

Figure 7 visualizes the training and validation loss over the training epochs as well as the Cohen's kappa on the validation sample. The results suggest that the cross-entropy loss on validation sample reaches a minimum already after 3 epochs and does not improve with further training. At the same time, kappa continues to increase up to the 15th epoch. Since we use kappa to evaluate the quality of our solution, we save model weights after 15 epochs.

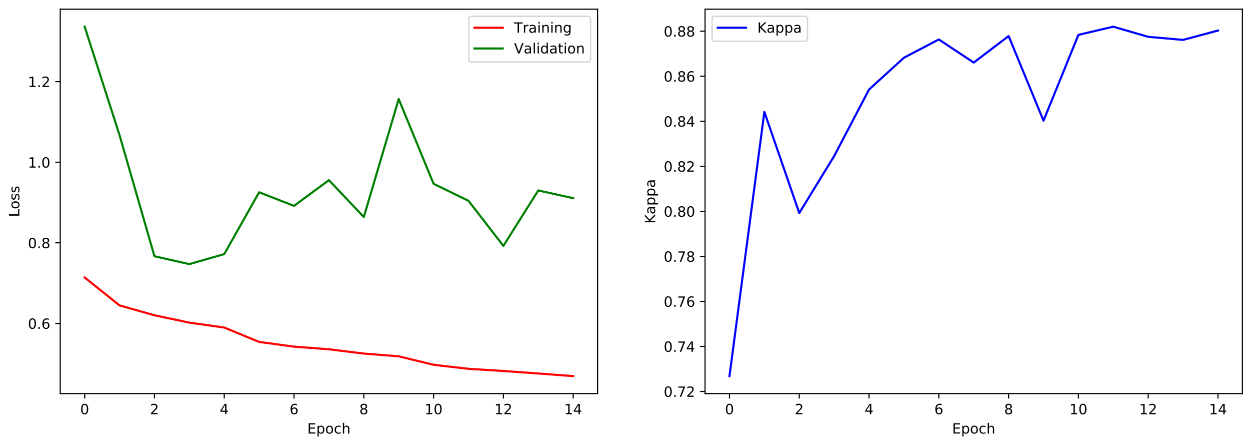

Figure 7. Pre-Training: Validation Performance Dynamics

Figure 8 visualizes the confusion matrix of the trained model on validation set. The numbers in the table cells are percentages. According to the results, the model does a poor job in distinguishing the mild and moderate stages of DR: 86% of images with mild DR are classified as moderate DR by the model. The best performance is observed for classifying the retina photographs of healthy people. Overall, we see that the model tends to confuse nearby severity stages but rarely missclassifies the proliferate and mild stages.

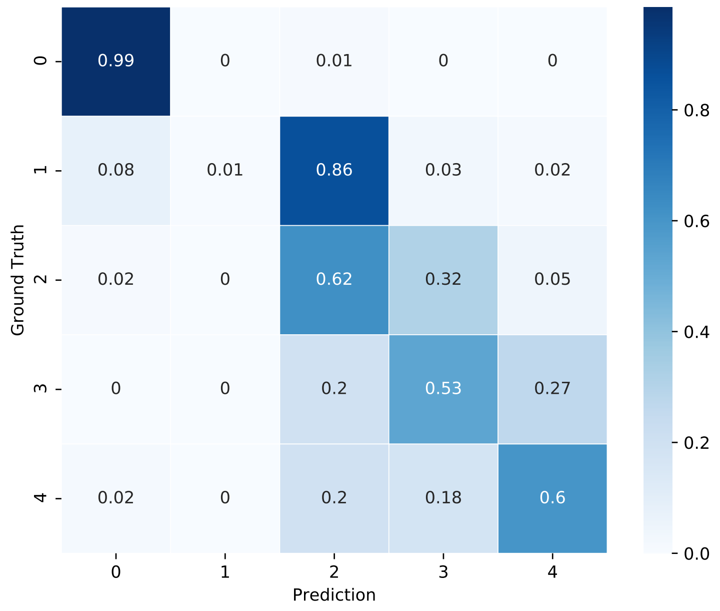

Figure 8. Pre-Training: Confusion Matrix

3.2.2 Fine-tuning

The fine-tuning on the main data set is performed using cross-validation with 4 folds. To ensure that we have enough examples of each class in the training sample, we perform cross-validation with stratification. On each iteration, we instantiate an EfficientNet B4 model with the same architecture as described in the previous section. Next, we load the saved weights from the model pre-trained on the supplementary data set. We freeze all network layers except for the last fully-connected classification layer. The weights in this layer are fine-tuned during training.

As on the pre-training stage, we use Adam optimizer with a learning rate of 0.001 and implement a learning rate scheduler which multiplies the learning rate by 0.5 after every 5 epochs. We also track the model performance on the validation fold and stop the training if kappa does not increase for 5 consecutive epochs. This process is repeated for each of the 4 folds, and the best model weights are saved for each combination of the training folds.

After running the cross-validation loop, we aggregate the model performance including validation loss and kappa over all out-of-fold examples from validation folds. This provides the estimate of the model performance. Based on the obtained kappa values, we test multiple variants of the model and iteratively refine the proposed solution. The details on the refinement process are provided in Section 3.3.

The results indicate that after pre-training, the model converges quickly during fine-tuning. The best validation performance is obtained after 3 to 7 training epochs depending on a fold. Figure 8 visualizes the confusion matrix of the best model after fine tuning. The confusion matrix illustrates the advantages of the fine-tuned model over the pre-trained CNN and indicates a better performance in classifying mild stages of the DR. However, we also observe that the model classifies too many examples as a moderate stage of DR (class = 2). We try to address this issue on the inference stage by manipulating the thresholds.

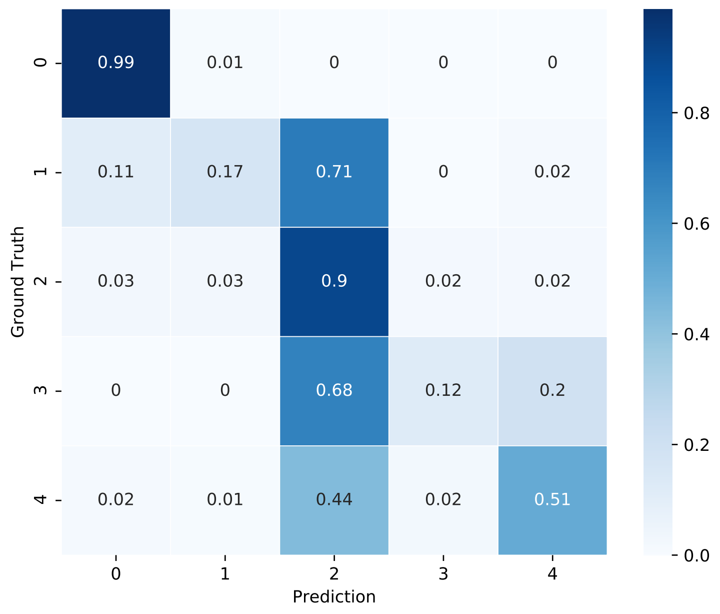

Figure 9. Fine Tuning: Confusion Matrix

3.2.3 Inference

The model predictions for the test sample are obtained by aggregating the predictions from the base models trained during the cross-validation loop. To do so, we extract class scores from the last fully-connected layer and define class predictions as the classes with the maximal predicted score out of 5 classes. Next, we average predictions of the 4 EfficientNet models trained on different combinations of the training folds. We also use test-time augmentations by creating 4 versions of the test images with random augmentations (horizontal and vertical flips) and average predictions over the two image variants. The final prediction is therefore the average of 4 models $\times$ 4 test image variants.

To reduce the number of examples classified as moderate DR (class = 2), we change thresholds used to round the averaged predictions into classes. We use the following vector of thresholds: $[0.5, 1.75, 2.25, 3.5]$: i.e., the final prediction is set to zero if the average value is below 0.5; set to one if the average value lies in $[0.5, 1.75)$, etc. This reduces the share of images classified as moderate DR in the test sample.

3.3 Refinement

The initial model is EfficientNet B4, where weights are initialized with the values pre-trained on ImageNet and no image preprocessing beyond resizing is applied. Fine-tuning this model on the 2019 data set by training the last classification layer achieves the average cross-entropy loss of 0.7135 and kappa score of 0.7696 on the validation folds. When applied to the test sample, the initial model achieves kappa of 0.7528, which is substantially worse compared to the benchmarks.

During the work on the project, we iteratively test different refinements and track the model performance within cross-validation to make decisions about including certain modifications. The empirical experiments include:

- Tuning the parameters of image preprocessing functions. This includes cropping functions and smoothing functions, data augmentations and image size. The results of the empirical experiments suggest that cropping images and using augmentations is crucial to achieve a better predictive performance.

- Testing three different EfficientNet architectures: B0, B4, and B7. The B4 architecture demonstrates the best performance and selected as a final model architecture. At the same time, model B0 demonstrates the fastest training and inference, which might be useful when deploying a model for screening new retina photographs. Model B7 might be able to achieve a better performance but its convergence speed is slower. Due to the resource constraints, it was only possible to train B7 with a smaller batch size.

- Testing unfreezing of different layer combinations. We find that the best performance is achieved when all model weights are optimized during the pre-training stage and only the last classification layer weights are optimized during the fine-tuning stage. Unfreezing more layers during fine-tuning does not improve the performance as the sample size is too small.

- Selecting the optimizer and the learning rate. We try two optimization algorithms: Adam and stochastic gradient descent (SGD). According to the results, using Adam results in a better performance. In addition, we find that using a learning rate scheduler by iteratively decreasing the learning rate after every 5th epoch provide a slight performance improvement.

The final model is selected based on the out-of-fold Cohen's kappa score. The best model achieves a kappa score of 0.8760 on the validation folds.

4 Results

4.1 Model Evaluation and Validation

As detailed above, we use cross-validation to train and validate a set of 4 EfficientNet B4 models. The validation performance is aggregated across the validation folds to provide a more reliable estimate of the model performance. We evaluate a model using both cross-entropy loss and the Cohen's kappa metric.

The final performance evaluation is conducted on the test set that was not included in the training data. The labels of the test images are not explicitly provided, but the model performance on this data can be evaluated using the «late submission» functionality on the APTOS 2019 Blindness Detection competition website.

The final CNN model achieves a cross-validation kappa score of 0.8760. When applied to the test images, the model demonstrates the kappa score of 0.8867, which is close to the cross-validation performance of the model.

4.2 Justification

The score achieved by our solution is higher than that of 2 out of 3 benchmark algorithms considered in the project. Despite the fact that our solution does not beat one of the benchmarks, our model scores in the top 50% of the 2,931 solutions developed by different teams during the APTOS 2019 Blindness Detection competition. We have intentionally selected a strong benchmark to aim at a higher performance level. Overall, our results demonstrate a competitive performance of the developed image classification solution but also provide opportunity for further improvement of the model.

There are multiple ways to further improve the predictive performance. First, employing a larger neural network architecture and extending the number of training epochs on the pre-training stage has a high potential for demonstrating a better performance. The strongest benchmark uses EfficientNet B5 architecture, which is deeper compared to the B4 model used in our solution. At the same time, this direction would require more computing power, which might not be optimal when considering the possible use of the automated retina image classification in practice.

Second, image preprocessing can be further improved. During the refinement process, the largest performance gains were attributed to including certain preprocessing steps when working with image data. This is a more efficient way to further improve the predictive performance.

Finally, the best solutions of other teams in the corresponding Kaggle competition rely on model ensembles trained on different image sizes and/or model architectures. Incorporating multiple heterogeneous models and blending their predictions could also improve the predictive performance of the proposed solution.

References

[1] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[2] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[5] RL Thomas, S Halim, S Gurudas, S Sivaprasad, and DR Owens. Idf diabetes atlas: A review of studies utilising retinal photography on the global prevalence of diabetes related retinopathy between 2015 and 2018. *Diabetes research and clinical practice*, page 107840, 2019.

[6] Tien Yin Wong and Neil M Bressler. Artificial intelligence with deep learning technology looks into diabetic retinopathy screening. *Jama*, 316(22):2366–2367, 2016.