APPLIED MACHINE LEARNING SYSTEMS II (ELEC0135 22/23) REPORT

SN: 19117737

ABSTRACT

Diabetic Retinopathy (DR) is one of the major causes for vision loss and blindness across the world. The lack of treatment can lead to unclear vision, blindness, and bleeding. An automated tool for screening could help detect DR in time for early treatments and could improve the quality of life of affected patients. This paper proposes a DR Risk classifier using Convolutional Neural Networks (CNN) trained on the APTOS 2019 Blindness detection dataset from Kaggle. An image pre-processing pipeline was developed to crop uninformative areas and filter the extensive noise introduced by real world circumstances. Multiple CNNs were trained, evaluated, and compared using 60/20/20 split whilst leveraging image augmentations, hyperparameter tuning, and transfer learning to enhance performance. The best performing model was RegnetY16GF with a score of 73.3% accuracy on the unseen dataset which is comparable to other works from the literature review and competitors from Kaggle given the computational limitations and the dataset that is by a factor of 10 smaller. The confusion matrix of the predictions showed high correlation between neighboring values. When implementing further improvements outlined in the future work section this tool is expected to reliably assess DR risk in patients. [1]

Index Terms – Computer vision, Convolutional Neural Networks (CNN), Diabetes, Blindness detection, Retinal photography, Transfer learning, Diabetic Retinopathy

1. INTRODUCTION

The use of computer vision in medical imaging has become increasingly popular over the past few decades [1]. Deep convolutional neural networks (CNN) have demonstrated remarkable performance in various fields, such as tumor detection, tumor segmentation. This paper investigates the use of CNNs in the risk assessment of Diabetic Retinopathy (DR). DR is a complication arising from prolonged exposure to high blood glucose levels in individuals with diabetes. For patients who had diabetes for more than 20 years, nearly all individuals diagnosed with type 1 diabetes and over 60% of those with type 2 diabetes manifest some degree of retinopathy [2], a pathological condition affecting the retina.

The retina is reliant on a consistent supply of blood, which is facilitated by a complex network of minuscule blood vessels. Prolonged high blood sugar level can disrupt the integrity of these blood vessels, leading to potential blindness if left undiagnosed and untreated. In the early stages, diabetic retinopathy may not exhibit any obvious warning signs, and its progression to a vision-threatening stage often requires several years. Early detection of ocular abnormalities through regular screening and prompt implementation of lifestyle modifications and/or interventions can effectively prevent or minimize further worsening or progression.

Among the established screening methods, retinal (fundus) photography with manual interpretation has gained wide acceptance as a reliable tool for detecting diabetic retinopathy, and its performance may surpass that of traditional in-person eye examinations. Figure 1 below depicts an example of a retinal image with severe DR highlighting the areas of interests when inspecting retinal photos.

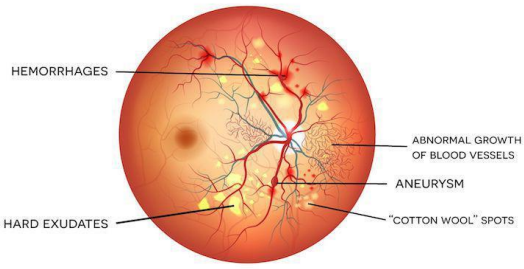

Figure 1 - Diabetic Retinopathy Example. Source: [3]

The training dataset used in this study consists of 3662 similar retina images with corresponding labels. Like typical real-world data sets, the data exhibits inherent noise in both the images and labels. The images were acquired from diverse clinical settings using a range of cameras, and were collected over an extended duration of time, thus introducing additional variability to the data set. Images contain artifacts, are out of focus, under- or overexposed. The images are labelled on a scale of 0 to 4 with the score denoting the severity of DR: 0 no DR, 1 mild, 2 moderate, 3 severe and 4 proliferative DR.

This paper aims to develop an automated tool for grading the severity of DR based on retinal photography using CNNs, which could accelerate the detection and treatment of DR. Given the significant noise in the dataset, first, an image pre-processing pipeline was developed to counter the variability in the dataset. This was followed by the development of multiple CNN models, during which image augmentation, transfer learning and hyperparameter tuning were performed to maximize performance.

The following sections of this paper give an exhaustive review of the existing literature, offering valuable insights into the field. Subsequently, a detailed description of the models employed in the study is provided, including their implementation and the experimental results. Finally, the conclusions drawn from the findings are summarized, and potential directions for future research and enhancement are proposed.

2. LITERATURE SURVEY

The manual diagnosis of medical images is an overwhelming and burdensome task, necessitating a revolutionary approach. Researchers worldwide have proposed numerous Machine Learning (ML) models to address this challenge, which can be categorized into two subgroups: traditional ML and deep learning methods. Various traditional ML approaches were introduced in the past using models support vector machines (SVM) [4], k-nearest neighbor (KNN) [5], Bayesian

[1] The GitHub repository containing the code for this project can be found under: https://github.com/TheSalDave/AMLS_II_assignment22_23

Classifier [6] among others in combination with DR feature extraction algorithms such as Principal Component analysis (PCA) [7], Gabor filter incorporated with Hough transform [8]. However, these models have shown poor generalization capabilities and are highly sensitive to noise in the dataset as shown in [9] which critically evaluates hundreds of ML studies. This has resulted in researchers shifting their focus towards deep learning methods in the past decade.

Deep learning approaches can be further subcategorized into DL&ML Hybrid, CNNs and Transfer learning.

DL&ML methods use CNNs to extract DR features which are then fed into traditional ML models for classification. An example of this was proposed by Kojasteh et al [10] They applied a patch and image based technique to detect signs of DR. They developed a CNN with 10 layers to identify regions of interest and feed the probabilistic output of the softmax layer into a SVM to identify them as exudates, hemorrhages or microaneurysm. Results have shown overall accuracies in the 90% range on their dataset and reduced the error rate in comparison to other works.

Xu et al. [11] have proposed a DL&ML hybrid by connecting CNNs with the ML algorithm XGBoost. For comparison, they have also developed 2 further models with multilayer perceptrons one of which used data augmentation techniques whilst the other did not. Results for all three options were in the 90% range but the data augmentation model scoring the highest.

CNN approaches apply deep learning models for feature extraction and for classification too. Wang et al. [12] trained CNNs on a large DR dataset and proposed 3 models using different image sizes: small 128x128, medium 256x256 and large 512x512 pixels. They have shown that increasing the image size enhances the performance of the models but also the computational costs of the trainings.

Another CNN approach was proposed by Orlando et al. [13]. They applied multiple image pre processing steps such as Gaussian filtering and morphological operations to reduce noise in the dataset. The paper combined the prediction of multiple models using ensembling and has shown that the results are better than individual models.

Transfer learning methods leverage pre-trained models to maximize the performance of CNNs. A comperative study of pre-trained models was performed by Sarki et al. [14]. They compared 13 pre-trained model architectures with Resnet50 showing best performance by retraining the classification layers starting at layer 100. They used fine tuning, data augmentation and dataset volume increase to maximize the performance and achieved 86% accuracy.

Hattiya et al. [15] presented a comparison of DenseNet201, InceptionV3, MobileNet, MnasNet, NASNetMobile , AlexNet and ResNet50. The study uses the same Kaggle 2019 APTOS dataset extended by others with the combined dataset consisting of 23513 retina images. A comprehensive evaluation process has shown that AlexNet performed the best with an accuracy of 81.32% on the training dataset.

Bodapati et al. [16] uses the same extended dataset including Kaggle 2019 APTOS dataset to train multiple model architectures. This paper compares various pooling techniques and concludes that averaging pooling performs the best with a maximum accuracy of 84.31%.

Overall, we have seen multiple approaches to solving the DR detection challenge. Traditional ML methods have shown poor generalization capabilities and high sensitivity to noise and will thus not be explored. The focus of this paper will be on deep learning techniques inspired by the literature review with a strong emphasis on transfer learning methods.

3. DESCRIPTION OF MODELS

This section describes the models that are used in this study. It gives an in-depth analysis of CNNs and its layers followed by an introduction to transfer learning. Finally, the model selection criteria is described explaining the rationale behind the choices made in this study.

3.1. Convolutional Neural Networks

Classical neural networks, also known as feedforward neural networks, have several limitations when it comes to image classification. They are computationally inefficient as they require many parameters to be learned, which can result in slow training times. Furthermore, they offer poor generalization, as they lack the ability to capture the underlying patterns in the data. Each image pixel is treated as an individual value and connected to every neuron in the network, which makes the network unable to understand the spatial relationships between pixels. This results in poor performance for images that are translated or shifted from the training data.

Convolutional Neural Networks (CNNs) address these limitations by introducing the concept of convolution, which is the process of applying filters or kernels over an image to extract relevant features. CNNs can efficiently learn to recognize patterns in images using local receptive fields and weight sharing. CNNs consist of multiple layers, each with a specific function. An example CNN can be seen in Figure 2.

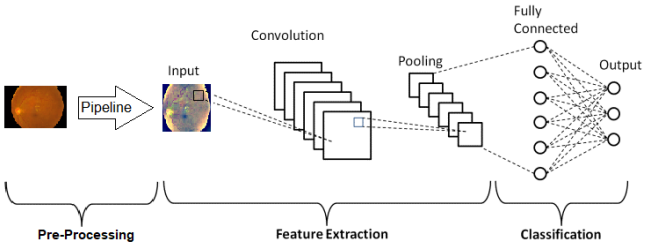

Figure 2 - Example CNN for DR classification (adapted from [9])

The convolutional layer applies filters to the input image to extract features, the pooling layer reduces the spatial dimensions of the feature maps to reduce computation, and the activation functions introduce non-linearity to the model. The final layers of a CNN are fully connected layers at the end for classification. An in-depth analysis of the CNN specific layers can be found in the following subsections.

3.2. Convolutional Layers

Convolutional layers are key components of CNNs. They take the input from the images as 2 dimensional matrices whilst maintaining spatial relationships of adjacent pixels. The architecture of a convolutional layer can be seen on Figure 3:

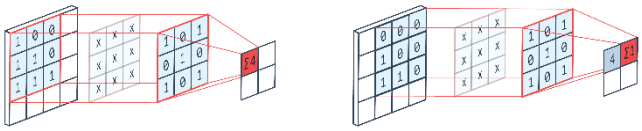

Figure 3 - Convolutional layer (Source: [12])

The output, A, can be calculated using equation 1:

[Eq. 1] $$A_j = f\left(\sum_{i=1}^{N} I_j * K_{i,j} + B_j\right)$$

Where the input layer, I, is convoluted with K, the kernel to which then a bias, B, is added. Subsequently, a non-linear activation function, f, is applied which gives the results for the output matrix A.

Each kernel serves as a local feature extractor that captures regional features from the input image. The goal of the learning process is to find kernel matrices that can effectively extract patterns for the specific image classification challenge. The kernel matrices and bias of the convolutional layer can be optimized using the backpropagation algorithm. This allows the CNN to adapt the kernel matrices during training to improve the quality of extracted features, leading to enhanced performance in image classification.

There exists a large variety of convolutional layers with different kernel sizes, strides, padding and activation functions. Kernel size determines the size of the kernel matrix, stride denotes the number of pixels the kernel is shifted in every move, padding represents the additional pixels added around the image and the activation function is used to add non-linearity to the model. All of these can be viewed as hyperparameters of the model that can be experimented with for better results.

3.3. Pooling layers

The pooling layer plays a significant role in reducing the dimensions of the features extracted by the convolutional layer. The pooling algorithms are applied to combine neighboring elements in the feature maps, which helps in capturing the most relevant information while reducing the spatial resolution. Two commonly used pooling algorithms are max- and average-pooling which can eb seen on figure 4:

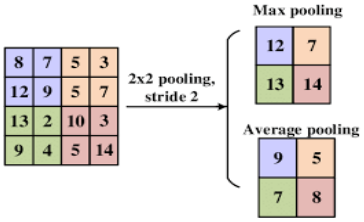

Figure 4 - Pooling layer explanation

Max-pooling involves selecting the maximum value from a group of neighboring elements in the feature map. It helps in preserving the most prominent features with the highest activation value, which can be critical for accurate object localization and recognition. Average-pooling calculates the average of the values in the group, providing a more generalized representation of the features.

CNNs can effectively reduce the spatial dimensions of the feature maps using pooling algorithms, which can help in reducing computational overhead, mitigating the risk of overfitting, and capturing more abstract and higher-level features.

3.4. Manual CNN

As part of this study a manually developed CNN was trained on the dataset. The input image was segmented into RGB layers and fed into the model. Thus, the first convolutional layer had 3 input channels. The number output channels were chosen to be 6 with a kernel size of 3 and stride of 1 without padding and a RELU activation function. This was then fed into a max-pooling layer with kernel size 2 and stride 2. The second convolutional layer and pooling layer pair were identical but with 6 input channels and 16 for output. Inspired by the literature review [12] the image size was chosen to be 256x256 pixels to achieve high performance within reasonable computational time. After passing through the convolutional and pooling layers the output becomes 62 pixels as can be seen in equation 2:

[Eq. 2] $$(((256-2)/2)-2)/2 = 62.5 \rightarrow 62$$

Since each convolutional layer decreases the matrix by 2 and each pooling layer halves the number of pixels. Thus after reshaping the output from the 16 output layers we feed it into a feedforward classifier layer with 16*62*62 input neurons and an arbitrarily chosen 256 output neurons. The second classification layer has 256 input neurons and 64 for output. The final layer with 64 input and 5 output neurons is then fed into a logarithmic SoftMax function which gives the probability of the output classes.

Parameters, architecture, and layer types of this CNN are chosen arbitrarily and serve only as an experiment for comparison purposes with pre-trained models.

3.5. Transfer learning

Transfer learning leverages the knowledge learned from large datasets and complex architectures of pre-trained models and applies it to the specific task of DR classification. The pre-trained models have learned to classify various objects from massive datasets and these features can be fine-tuned and combined with DR-specific data. Generally speaking, lower layers capture low-level features such as textures and edges, while higher layers capture more abstract and complex features, such as objects and shapes. By leveraging pre-trained weights, transfer learning can mitigate the limitations of small datasets, leading to improved classification accuracy and generalization performance. Additionally, transfer learning is of an immense advantage in scenarios where resources are limited, or time constraints are critical as it reduces the training time and computational resources required to train a CNN from scratch.

3.6. Selection criteria

When making design choices or selecting models, it is crucial to consider the constraints of limited computational resources. The balance between accuracy and computational time should be thoroughly evaluated at each decision point.

Furthermore, prioritizing the demonstration of machine learning techniques is also important.

When assessing pre-trained models trained on the ImageNet dataset to classify 1000 unique classes, the following parameters were considered:

- Acc@1: Measures the percentage of correct predictions made by the model when considering only the most confident prediction.
- Acc@5: Measures the percentage of correct predictions made by the model when considering the top 5 most confident predictions. This allows for a bit more flexibility, as a correct prediction can still be counted even if it's not the absolute top prediction.
- Params: This represents the number of parameters of the model. Parameters are the learnable weights and biases that are optimized during the training process to make predictions. The total number of parameters in a model can give an indication of its complexity and memory usage.
- GFLOPS: Represents the number of billions (Giga) of floating-point operations per second that the model can perform. It provides an estimate of the computational power required to execute the model during inference, which can be useful for understanding the model's efficiency and required computational resources
- Literature review: Performance on DR datasets from the literature review and Kaggle competitors

The selected models for this study are: Manual CNN, RegnetY16GF, ResNet50, EfficientNetB7 and AlexNet.

4. IMPLEMENTATION

The implementation of this work was carried out using Python, leveraging several external libraries. For visualization and data analysis popular tools such as NumPy, OS, Matplotlib, Pandas were utilized. For deep learning models, the PyTorch library, including PyTorchVision's CNN-specific tools, were employed. Additionally, the development process involved utilizing Keras_preprocessing, OpenCV (cv2), and fastai libraries.

The following subsection provide a detailed implementation of the models. They give an in-depth analysis of each process of the model development and the applied ML techniques. The first subsection showcases the exploratory data analysis (EDA) of the dataset and the development of the preprocessing pipeline This is followed by the second subsection that described the steps of developing the classifier model.

4.1. Pre-Processing pipeline

Data pre-processing is a crucial first step in image classification projects, as it ensures that the data is properly formatted, of high quality, and meets the requirements of the subsequent modelling algorithms. The following subsections provide an overview of the EDA steps conducted on the dataset, as well as the development of the image pre-processing pipeline and the application of image augmentations to the training dataset.

4.1.1. EDA

The dataset used in this study was obtained from the 2019 APTOS Kaggle competition. It comprised of 3662 labeled .png images of retinas. Similar to typical real-world datasets, this dataset exhibited inherent noise in both the images and labels. The images were acquired from diverse clinical settings using various cameras and were collected over an extended duration of time, introducing additional variability to the dataset. The images displayed artifacts, were sometimes out of focus, and could be under- or overexposed. The severity of diabetic retinopathy (DR) in the images was labeled on a scale of 0 to 4, with the scores denoting the severity level as follows: 0 for no DR, 1 for mild, 2 for moderate, 3 for severe, and 4 for proliferative DR. The frequency of the labels can be inspected on Figure 5 below:

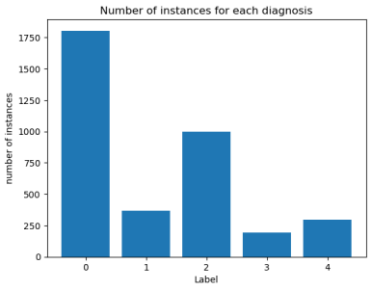

Figure 5 - Frequency of labels in the dataset

As can be seen on the figure the dataset is imbalanced with significantly more samples for no DR. There are three approaches to dealing with this issue. Oversampling the minority classes could balance the dataset, but it would create artificial images from existing ones that could lead to overfitting. Furthermore, it would increase the dataset and thus increase the computational time whilst not providing and new unique information. Another approach would be to under sample the majority class. This would however lead to loss of information on the already small dataset.

The chosen approach was to apply cost sensitive learning where different misclassification losses are associated to different classes. This method preserves all the information in the dataset and does not affect computational cost negatively. It also offers the potential for clinicians to tune costs and assign higher losses when misclassifying severe cases to decrease the occurrence of false negatives.

When inspecting the sizes of the images it was found that they range from 358x474 to 2848x4288 pixels with various size ratios.

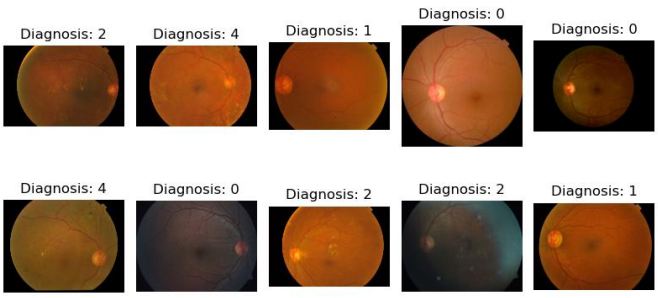

Figure 6 - Example set of original images

Figure 6 illustrates a sample set of images along with their corresponding diagnoses. The images exhibit variability in size, lighting conditions, and clarity of relevant information. Some images, such as 1, 2, and 3 in the first row, and 1, 3, and 5 in the second row, contain irrelevant black areas that could be cropped. Additionally, most images in the first row, except for 4 and 5, have cut-off top and bottom circular areas. There is also a wide range of color palettes and lighting conditions, with images 5 in the first row and 2 and 4 in the second row being particularly dark. Clearly, pre-processing of these images is necessary to reduce noise and improve model performance.

4.1.2. Image Pre-processing pipeline

The first stage of the pre-processing pipeline is to crop uninformative areas of the image as demonstrated in the previous section. For this, a function was developed that converts the image into greyscale. Then, given a specified threshold below which pixels are considered black, a mask is created. When applying this mask to the image the uninformative areas are cropped.

The next step of the pipeline is to resize the images as all input images need to be of the same size. For this, size 256 was chosen inspired by [12] expecting high accuracy for reasonable computational time. Another advantage of choosing this size is that all images are scaled down thus not creating artificially enlarged blurry images.

The final step of the pipeline is to deal with the high variability of colors and lighting conditions. For this OpenCV library was used. A Gaussian filter was applied with a sigma value of 10 that was found by manual experimentation. The resulting images after the pipeline was applied to previous set of images can be seen on Figure 7.

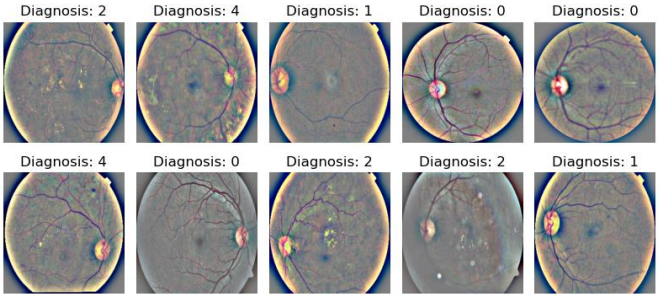

Figure 7 - Example set of images after pre-processing pipeline

As depicted in the figure, the images exhibit similar conditions after undergoing the pre-processing pipeline. They have consistent color ranges, lighting conditions, and sizes, with similar cropping and aspect ratios, making the relevant information easily observable. Applying this pipeline to all images contribute to a better performance as shown by multiple papers in the literature review.

4.1.2. Image Pre-processing pipeline

Image augmentation is used to increase generalization capabilities of models by artificially enlarging train datasets using image transformations. One of these is flipping the image with a probability of 50%. A further method is to apply color jitter which adjusts the contrast, hue and saturation of the image within a specified range, here 0.2, 0.3 and 0.2 respectively. Random affine transformation is a further augmentation technique. It rotates the image by a random degree within a specified range, here 20 degrees, and applies a shear transformation to the image within a range, here 0.2. Finally, a random perspective transformation is applied that warps the image in a way that simulates perspective distortion. These transformations are applied on the training dataset before the pre-processing pipeline and the effects can be seen on the example set in Figure 8:

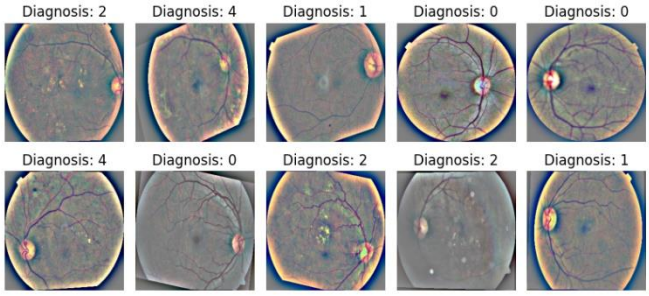

Figure 8 - Example set of images after data augmentation

Whilst the effect of some of the transformations on the images is minimized by the pipeline it introduces slight changes in pixel values that aids in the generalization capabilities of the model. The transformations were inspired by the literature review and competitors in the Kaggle competition. The specific values were obtained after manual experimentation.

4.2. Model development

The following subsections discuss the development steps of the models.

4.2.1. Dataset

For the development of the model, a custom dataset was developed that stores images and the corresponding labels. As part of the custom dataset the pre-processing pipeline is applied whenever accessing images. Whilst the Kaggle competition supplied a test dataset the images do not have corresponding labels. Thus, the dataset is split into train, validation and test sets in a ratio of 60/20/20. The various models are trained on 60% of the dataset and evaluated on 20%. The remaining 20% of the dataset is kept for testing purposes and will only be used on the final proposed model. The images are then loaded into PyTorch data loaders with specified batch sizes and transformations. For the training set the augmentations are applied first which is then fed into the pre-processing pipeline. For validation and test images only, the pipeline was applied.

4.2.2. Model training

For the pretrained models, the selection of weights is based on their performance on the ImageNet dataset, a widely used benchmark in computer vision research. While it is common to freeze only the early convolutional layers and fine-tune the subsequent layers along with the classification layers, due to constraints in computational resources, it was decided to freeze all convolutional layers and solely retrain the classification layers of the models.

Furthermore, the classification layers were adjusted to output a reduced number of classes, specifically 5, instead of the original 1000. This modification was implemented to tailor the model to the specific classification task at hand. In contrast, for the manual CNN, all parameters were trained from scratch, implying a comprehensive optimization of the network architecture and weights without any reliance on pre-existing weights or features.

The choice of optimizer for this study was ADAM, with the default learning rate set to 0.001. The Cross Entropy loss function was utilized as the objective function during model training. As discussed in the Exploratory Data Analysis (EDA) section, to effectively address the issue of class imbalance in the dataset, a cost-sensitive learning approach was employed. Specifically, the class weights, denoted as $\boldsymbol{w_i}$, were calculated based on the inverse frequency of the labels, using the formula presented in Equation 3.

[Eq. 3] $$\boldsymbol{w_i = \frac{total\ samples}{class\ frequency \times number\ of\ classes}}$$

The loss determined by the cross-entropy function is then applied by the corresponding class weights. Initially, all models were trained for 10 epochs with a batch size of 32. Depending on the results of this initial run, hyperparameter optimization was applied to determine the best model.

4.2.3. Hyperparameter tuning

There is a nearly unlimited combination of hyperparameters for the final model. One of the hyperparameters that can be tuned is the batch size which determines the number of samples used in each training iteration. Larger batch sizes lead to faster training times and more stable updates while smaller batches result in more frequent updates but with more noise. This is good for generalization; however, it can increase convergence time significantly. Common batch size values are multiples of 2 and for our dataset in the range of 32/64. Learning rate determine the step size of the optimizer when updating weights during training. Too high values result in poor convergence while too low values lead to slow training. Common values are multiples of 10. The number of epochs can also affect the results. If they are too little, the training will not converge, if they are too high the model will overfit the data. An example for this can be seen on Fig:

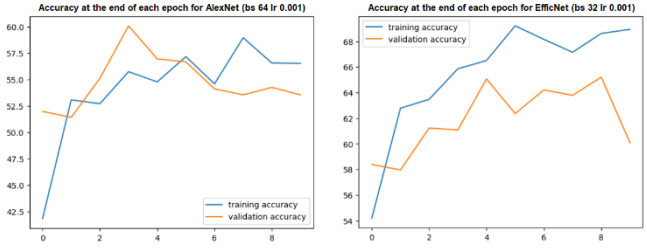

Figure 9 - Overfitting example learning curve

As identified in the literature review, the input image size also affects the performance of the model which can be thought of as a further hyperparameter of the model. Another parameter is the optimization algorithm. Common ones are Stochastic Gradient descent (SGD), Adam or RMSprop. Different model architectures with different amount of channels/neurons/number of layers/types of activation functions. For pre-trained models we can also adjust the number of layers to be retrained or specific layers within the architecture. Given our imbalanced dataset we can also adjust the class weights according to results. The list of hyperparameters to optimize is endless

The hyperparameters can be optimized using a grid search that tries the different combinations of the hyperparameter options. A common platform to perform hyperparameter sweeps is Weights and Biases (WandB). Given the computational limitations of this study this was not possible. To tune hyperparameters a manual approach was applied. First all models were trained using the parameters described in the previous section. Models with good performance were then retrained using a batch size of 32. The better performing hyperparameter was then used to try different learning rate values. Inspired by the learning curves when training the models, it could be seen that the models reach high performance quickly and diverge from it later which indicated that learning rates were too high. To mitigate this issue two further options were tried besides the original value. One of these was to decrease it by a factor of 10 to 0.0001. The other approach was to mix learning rates by applying a learning rate scheduler. Every 3 epochs the learning rate decreases by a factor of 10. To select the number of epochs, the learning curves were inspected to avoid overfitting like shown in Figure 9.

5. EXPERIMENTAL RESULTS AND ANALYSIS

This section presents the experimental results on unseen dataset and lists directions for future research.

5.1. Analysis of results

The experimental results of this study can be found in table 1 below. It is interesting to observe how transfer learning and hyperparameter tuning results in a notable performance increase. The best performing combination was RegnetY16GF with batch size of 64 and learning rate scheduler. This model was then retrained on the train+validation datasets and used to diagnose unseen test images. The testing accuracy was 73.3% which is comparable to other works in the literature review and competitors from Kaggle which were in the 80% range for unseen images. The difference in performance can be explained by limited computational resources and the dataset that was smaller by a factor of 10.

Table 1 - Experimental Results

Model	Train Acc	Val Acc	Test Acc
CNN(bs 64 lr 0.001)	66.36%	62.50%	
AlexNet (bs 64 lr 0.001)	60.89%	60.09%	
Resnet50 (bs 64 lr 0.001)	46.32%	48.58%	
RegnetY16GF (bs 32 lr 0.001)	69.46%	70.74%	
RegnetY16GF (bs 64 lr 0.001)	71.65%	70.60%	
RegnetY16GF (bs 64 lr 0.0001)	72.61%	71.16%	
RegnetY16GF (bs 64 lr MIX)	75.37%	71.88%	73.3%
EfficientNetB7 (bs 32 lr 0.001)	69.14%	65.06%	
EfficientNetB7 (bs 64 lr 0.001)	69.53%	65.9%	
EfficientNetB7 (bs 64 lr 0.0001)	72.61%	71.16%	
EfficientNetB7 (bs 64 lr MIX)	66.89%	63.49%	

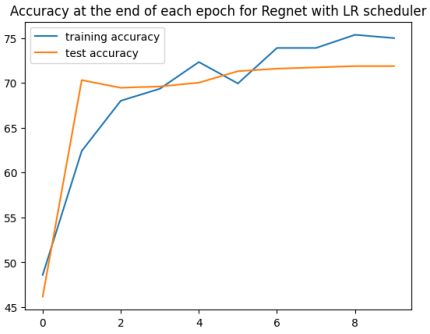

Figure 10 - Learning curve of the final model on the test set

Figure 10 depicts the learning curve of the final model. We can observe that the test accuracy has a very small increasing trend which suggests a potential for small improvements for longer training. To gain a deeper understanding of the results the predictions were compared with actual values using a correlation matrix as can be seen on figure 11:

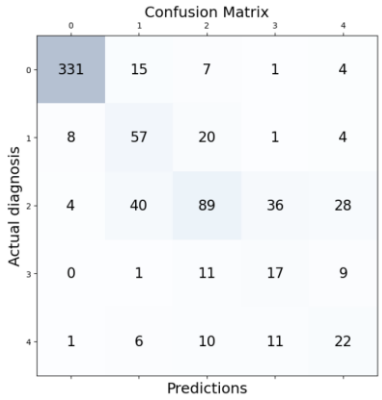

Figure 11 - Confusion Matrix of the results

It can be observed, that misclassified predictions are close to the diagonal thus the actual values. These small errors might be explained by the ambiguity of severity levels and the images being labelled by various clinicians. Furthermore, when the model predicted no DR it was more than 96% accurate with only 1.45% being significant false negatives. These values could be easily improved further by adjusting the class weights. Whilst the overall accuracy of the model might decrease the tool could be used to filter out no DR cases which would remove about half of the cases and save time for clinicians.

5.2. Future research

The performance of the model can be improved by performing a thorough grid search sweep using WandB for the hyperparameters listed in the hyperparameter tuning section. Exploring other pretrained models such as VGG-16, DenseNet and GoogLeNet could also lead to higher prediction accuracy. Furthermore, combining the small dataset with large external ones as it was done in the literature review and by Kaggle competitors, would certainly benefit the predictions.

Another approach is to combine the prediction of multiple models using ensembling techniques. The winner of the competition trained 8 various models and applied ensembling techniques to determine the prediction based on the ones from the 8 models.

A further technique that could lead to improved scores is pseudo labelling. The Kaggle competition provided a relatively large test set without any labels. Using the models to create labels for those images and retrain the models on the enlarged could lead to increased accuracy. The pseudo labelling can be applied multiple times iteratively and lead to better scores with every iteration.

Finally, assuming a reliable model has been developed explainability techniques such as saliency maps could be applied to analyze and segment the final model. This could improve the reliability and offers more insight into regions of interest when analyzing retinal images.

6. CONCLUSION

This paper proposed a DR Risk classifier using Convolutional Neural Networks (CNN) developed using the small APTOS 2019 Blindness detection dataset from Kaggle. An image pre-processing pipeline was successfully implemented to crop uninformative areas, resize images and filter the extensive noise introduced by real world circumstances. Multiple CNNs were trained, evaluated, and compared using 60/20/20 split of the dataset. A large variety of image augmentation techniques were implemented to artificially increase the dataset and improve the generalization of the models. To maximize performance hyperparameter tuning was performed which due to limitational computational skill had to be done manually. The work has shown how even manual hyperparameter tuning can lead to increased performance. The study also showcases the performance boosting effect of transfer learning. By training pre-trained models, namely, AlexNet, Resnet50, RegnetY16GF and EfficientNetB7. The best performing model was RegnetY16GF with a score of 73.3% accuracy on the unseen dataset which is comparable to other works from the literature review and competitors from Kaggle given the computational limitations and the small size of the dataset. The confusion matrix of the predictions showed high correlation between neighboring values. By exploring direction of future research, a tool can be developed that can reliably detect and assess the risk of DR and save millions of patients from blindness.

12. REFERENCES

[1] P. Dutta, P. Upadhyay, M. De and R. G. Khalkar, "Medical Image Analysis using Deep Convolutional Neural Networks: CNN Architectures and Transfer Learning," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, India, 2020.

[2] A. D. Association, "Diabetic Retinopathy," *Diabetes Care,* vol. 25, no. 1, pp. 90-93, 2002.

[3] "Ophthalmology Physicians & Surgeons, PC," [Online]. Available: https://www.eyeops.com/contents/our-services/eye-diseases/diabetic-retinopathy. [Accessed 06 04 2023].

[4] R. Priya and P. Aruna, "Diagnosis of diabetic retinopathy using machine learning techniques," *ICTACT JOURNAL ON SOFT COMPUTING,* vol. 3, no. 4, p. 563 – 575, 2013.

[5] E. BM, H. OK, L. OV, M. K, J. B, K. D and C. DA, "Screening for diabetic retinopathy using computer based image analysis and statistical classification.," *Comput Methods Prog Biomed,* vol. 62, no. 3, p. 165–175, 2000.

[6] H. HK, L. CC, Y. CY, K. SW and Y. SS, "A novel optic disc detection scheme on retinal images," *Expert Syst Appl,* vol. 39, no. 12, p. 10600–10606, 2012.

[7] G. R and B. L, "Automatic segmentation of blood vessels from retinal fundus images through image processing and data mining techniques," *Sadhana,* vol. 40, no. 6, p. 1715–1736, 2015.

[8] D. C, U. B, M. SS, B. S and W. T, "Parameter-free optic disc detection.," *Comput Med Imaging Graph,* vol. 35, no. 1, p. 51–63, 2011.

[9] D. D, B. S.K. and B. S., "A critical review on diagnosis of diabetic retinopathy using machine learning and deep learning," *Multimed Tools Appl,* vol. 81, p. 25613–25655, 2022.

[10] K. P, A. B and K. D.K., "Fundus images analysis using deep features for detection of exudates, hemorrhages and microaneurysms.," *BMC Ophthalmol,* vol. 18, no. 288, p. 25613–25655, 2018.

[11] K. Xu, D. Feng and H. Mi, "Deep Convolutional Neural Network-Based Early Automated Detection of Diabetic Retinopathy Using Fundus Image," *Molecules,* vol. 22, no. 12, p. 2054, 2017.

[12] W. Z and Y. J, "Diabetic retinopathy detection via deep convolutional networks for discriminative localization and visual explanation," in *The Workshops of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] O. JI, P. E, d. F. M and B. MB, "An ensemble deep learning based approach for red lesion detection in fundus images," *Comput Methods Prog Biomed,* vol. 153, p. 115–127, 2018.

[14] S. R, Michalska S, A. K, W. H and Z. Y, "Convolutional neural networks for mild diabetic retinopathy detection: an experimental study," bioRxiv, 2019.

[15] T. Hattiya, K. Dittakan and S. Musikasuwan, "Diabetic Retinopathy Detection Using Convolutional Neural Network: A Comparative Study on Different Architectures," *Mahasarakham International Journal of Engineering Technology,* vol. 7, no. 1, pp. 50-60, 2021.

[16] B. J.D., S. N.S. and V. Naralasetti, "Deep convolution feature aggregation: an application to diabetic retinopathy severity level prediction," *Signal, Image and Video Processing,* vol. 15, p. 923–930, 2021.