UNIVERSITY OF MORATUWA, SRI LANKA
Faculty of Engineering
Department of Electronic and Telecommunication Engineering
Semester 5 (Intake 2020)

EN3160 - Image Processing and Machine Vision
Diabetic Retinopathy Severity Grading

AMARASEKARA A. T. P.	200023C
THARUKA K.P.	200641T

Github Repository: https://github.com/thisariii01/Diabetic-Retinopathy-Severity-Grading

Table of Contents

Abstract

Diabetic Retinopathy (DR) severity grading involves the classification of retinal images into five levels of severity ranging from 0 to 4, representing No DR, Mild Non-Proliferative Diabetic Retinopathy (Mild NPDR), Moderate NPDR, Severe NPDR, and Proliferative Diabetic Retinopathy (PDR). With the advancement of Deep Learning techniques, recent studies have employed various architectures and models to grade these severity levels of DR, aiding timely diagnosis and treatment for diabetic patients. This report explores the recent works involved in grading the severity of DR and our approach to solving the problem using Image Processing and Machine Vision techniques, including the preprocessing of retinal fundus images, the training of a machine learning model, and assessing its performance metrics.

1. Introduction

Diabetic Retinopathy (DR) is an ocular complication of diabetes that affects the retina, the light-sensitive layer at the back of the eye. The retinal vessels, with its high metabolic demand, are vulnerable to damage caused by oxidative stress, which occurs in pathologic conditions like chronic diabetes. Diabetic Retinopathy (DR) is recognized as a microvascular disease, characterized by various retinal abnormalities, including vascular changes, haemorrhages, and fluid extravasation. Over time, these changes can lead to vision distortion and a reduction in visual acuity. It is a leading cause of vision impairment and blindness among individuals with diabetes.

DR is characterized by a range of abnormalities in the retinal blood vessels, which can progress through various stages of severity. These severity levels are classified into five distinct stages, each denoting different degrees of retinal damage (according to the international clinical DR disease severity scale).

No DR (No Diabetic Retinopathy): This stage indicates the absence of any apparent retinal abnormalities, making it the ideal condition for individuals with diabetes.

Mild non-proliferative diabetic retinopathy (Mild NPDR): In this stage, early signs of damage to the blood vessels are observed, such as microaneurysms.

Moderate NPDR: This stage features more widespread blood vessel abnormalities (including more than just aneurysms), however still less than severe NPDR.

Severe NPDR: This stage may include any of the abnormalities, Intraretinal haemorrhages (20 in each quadrant), definite venous beading (in two quadrants), Intraretinal microvascular abnormalities (in 1 quadrant).

Proliferative Diabetic Retinopathy (PDR): This is the most advanced and severe stage of DR. If one of Neovascularization or Vitreous/preretinal haemorrhage is present in addition severe NPDR abnormalities, it is classified into this stage.

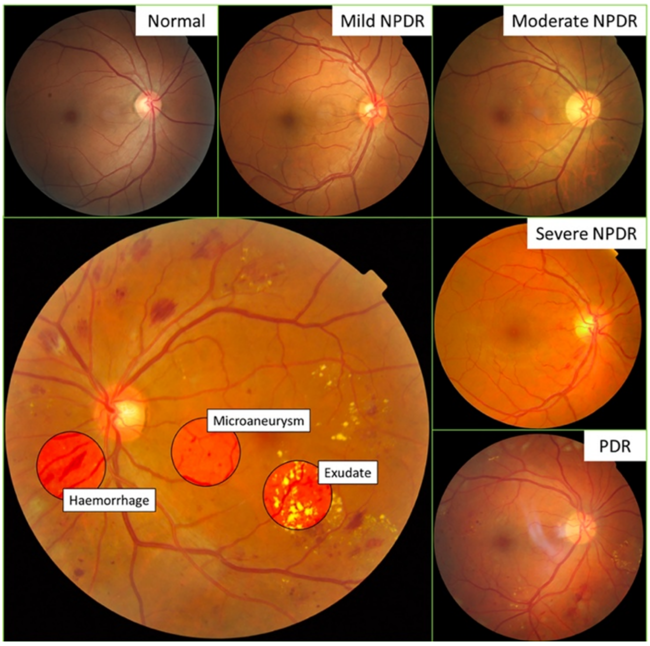


Figure 1 - Example of images that represents the severity levels and the presence of microaneurysm, haemorrhage and exudate in a severe non-proliferative diabetic retinopathy (NPDR) fundus image

These severity levels are critical for healthcare professionals to determine the appropriate treatment and management for individuals with diabetes. Regular eye screenings are essential to detect DR at an early stage when it is more manageable reducing the risk of vision loss in diabetic patients.
However, the increasing population of diabetic patients causes difficulty for ophthalmologists to conduct manual DR screenings routinely and provide timely diagnosis and treatments. This raises the need for automated DR screening systems. With the advancement of technology, this automation provides us with several advantages including increased efficiency, reproducibility and scaling and also as a method to verify the manual assessments reducing the errors occurred during the screening process.

2. Existing Works

Diabetic retinopathy severity grading through deep learning methods is widely adopted and extensively researched globally. Many studies focus on enhancing the accuracy of these models. The majority of these research efforts leverage Convolutional Neural Networks (CNNs) as the foundational architecture for their classification algorithms. While some studies employ custom-designed CNN architectures, others utilize established architectures like VGG and ResNet50. Remarkably, these studies have achieved notable accuracy levels, regardless of whether they employ custom or predefined architectures. This illustrates the versatility and success of CNNs in the field of diabetic retinopathy severity grading

3. Dataset

Background researches convey that several datasets are available in the internet that can be used for diabetic retinopathy severity grading such as EyePACS(Kaggle), APTOS(Kaggle), Messidor-2. Each dataset contains several images and our project we have used EyePACS(Kaggle) data set which actually consists of two data sets one is resized version and the other one is resized and cropped version. Cropped version is more suitable since it has removed many redundant data by cropping the image. The selected dataset consists of 35108 images but it is highly imbalance.

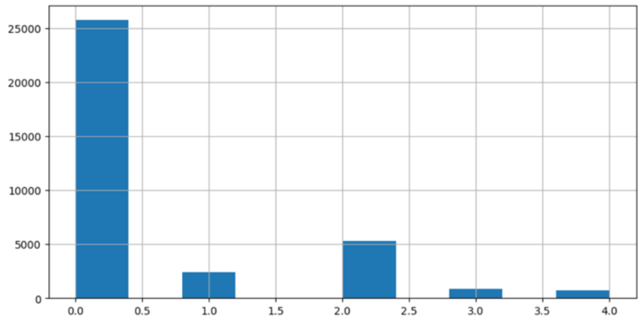


Figure 2 – Histogram of input images with respect to the severity levels

4. Preprocessing

Preprocessing is a crucial step in our project, and it involves two main stages, Image quality enhancement and Data Augmentation.

4.1 Image Quality Enhancement

The images we work with exhibit minor variations, primarily caused by differences in lighting conditions when capturing fundus images. To accentuate critical features and overcome this issue, we employ preprocessing techniques. In this stage, we focus on enhancing image quality.

To achieve this, we employ a method to sharpen the images and increase their intensity. This is accomplished by taking the weighted difference between the original image and a Gaussian-blurred version of the same image (i.e., A(Original Image) - B(Smoothed Image)), and then enhancing the image's intensity(Brightness and Contrast). This process helps bring out and highlight important details in the images.

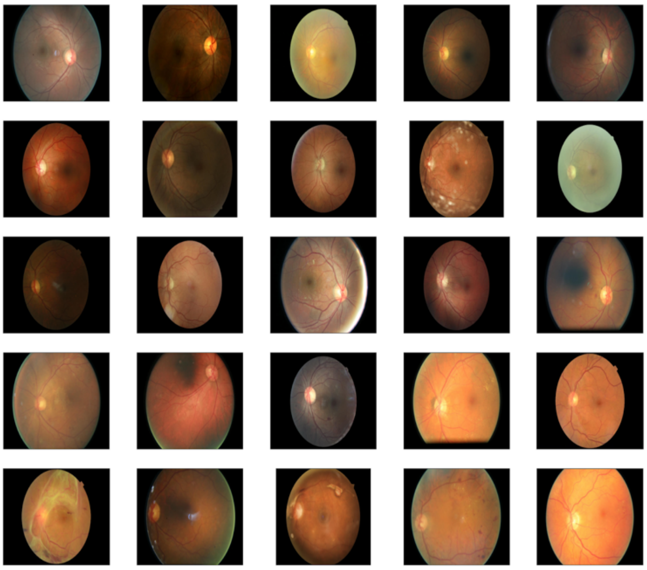


Figure 3 – Original image samples

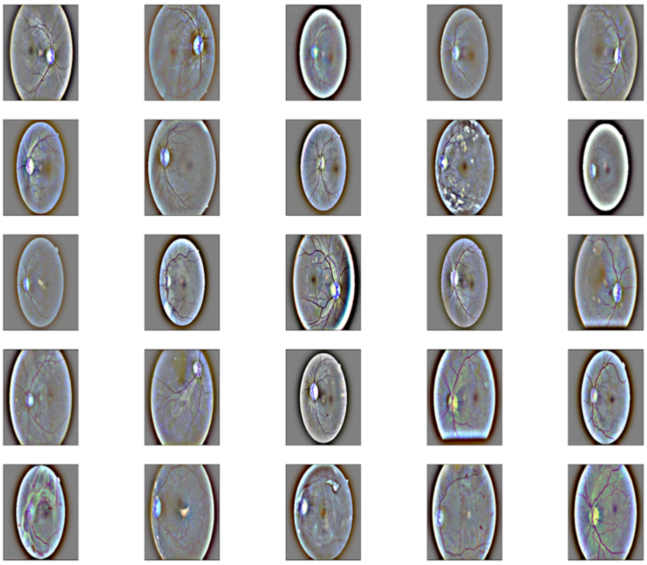


Figure 4 – Quality Enhanced image samples

4.2 Data Augmentation

To mitigate the risk of overfitting in our project, we employ data augmentation, a crucial technique for improving the robustness of our machine learning model. Data augmentation involves creating additional training samples by applying various transformations to the existing images. These transformations help the model become more adaptable and capable of generalizing well to new data.

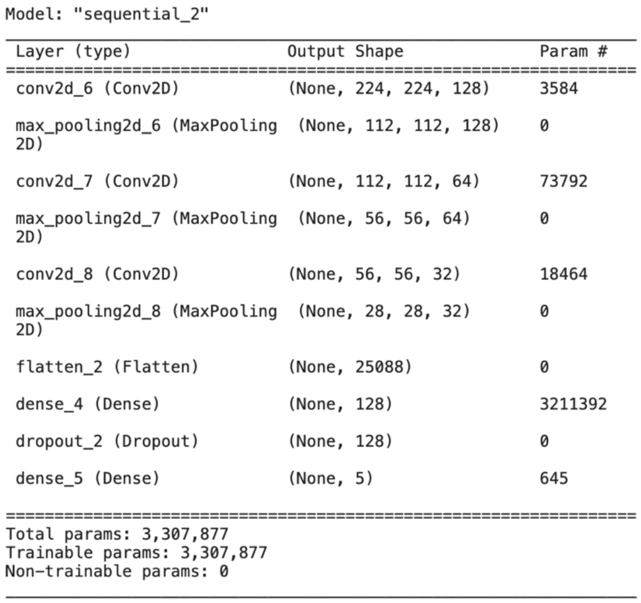
Key data augmentation techniques that we have used here is shearing, zooming, flipping , and rotation.

Data normalization, which involves rescaling the data to a common range, is a critical aspect of preparing data for neural network training. In our workflow, we have seamlessly integrated this normalization step with the data augmentation process. This ensures that the data, while being augmented to increase its diversity and robustness, is also consistently and appropriately scaled for optimal neural network performance. Normalization is vital for enhancing convergence, mitigating gradient-related issues, promoting model robustness, and supporting the generalization of our neural network to new data.

5. Model

Initial Custom Model

Initially, we designed a custom neural network comprising three convolutional layers with padding, each followed by max-pooling, and two dense layers with a dropout rate of 0.5. The input layer was configured to take RGB color images of size of (224,224). To optimize this model, we employed the Adam optimizer with varying learning rates (0.01, 0.03, and 0.1). For loss calculation during model compilation, we utilized Sparse Categorical Entropy. The dataset was split into subsets of 28,086 images for training, 3,511 images for validation, and 3,511 images for testing.



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 224, 224, 128)     3584

 max_pooling2d_6 (MaxPooling  (None, 112, 112, 128)    0
 2D)

 conv2d_7 (Conv2D)           (None, 112, 112, 64)      73792

 max_pooling2d_7 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 conv2d_8 (Conv2D)           (None, 56, 56, 32)        18464

 max_pooling2d_8 (MaxPooling  (None, 28, 28, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 25088)             0

 dense_4 (Dense)             (None, 128)               3211392

 dropout_2 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 5)                 645

=================================================================
Total params: 3,307,877
Trainable params: 3,307,877
Non-trainable params: 0
_________________________________________________________________
```

Figure 5 – Summary of the Initial Model

Pre-trained Models

However, we encountered a significant challenge with this model when dealing with a highly imbalanced dataset, which exhibited a substantial bias toward class 0 (Normal). The model consistently predicted class 0 for all input images, failing to learn and distinguish the features present in the data effectively. To address this issue, we attempted to balance the dataset by under sampling the class 0 data and oversampling the minority classes, utilizing replacement to augment the dataset with more data samples. Despite these efforts, the model continued to exhibit the same behavior of predicting a single class for all input data, regardless of the data's true features.

In response to these challenges, we decided to explore the potential of transfer learning with a pretrained model, specifically the VGG16 architecture. VGG16 is a well-established convolutional neural network known for its usage in image classification, with 16 layers.

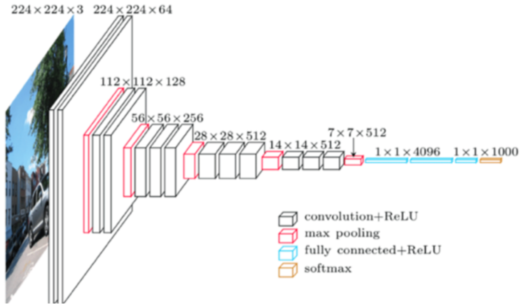


Figure 6 – Architecture of VGG16

This model is capable of classifying over 1,000 different classes with an accuracy of 92.7%. While we made some minor modifications to adapt the VGG16 model to our specific requirements and made attempts to enhance the validation accuracy, our results did not yield a significant improvement. Our conclusion was that, despite its effectiveness in classifying images with significant differences, the VGG16 model might not be sufficiently advanced to extract the intricate features present in our dataset of fundus images. We even explored various optimization strategies, including learning rate adjustments over epochs to prevent getting stuck in a plateau, but the results remained suboptimal.

As a result, to capture the detailed features within our data, we experimented with a more complex model, efficientnetB0. Unfortunately, the results did not meet our expectations. Consequently, we decided to design another custom model, one that could be easily trained, given the relatively small number of trainable parameters.

Finalized Custom Model

In the final step, we used a custom neural network which comprises convolutional layers , pooling layers , dense layers and dropout layers. It is much simpler than the pretrained models we used initially.

Our custom Convolutional Neural Network (CNN) is composed of the following key components.

Convolutional Layers: We've integrated four convolutional layers to capture essential features from the input data.

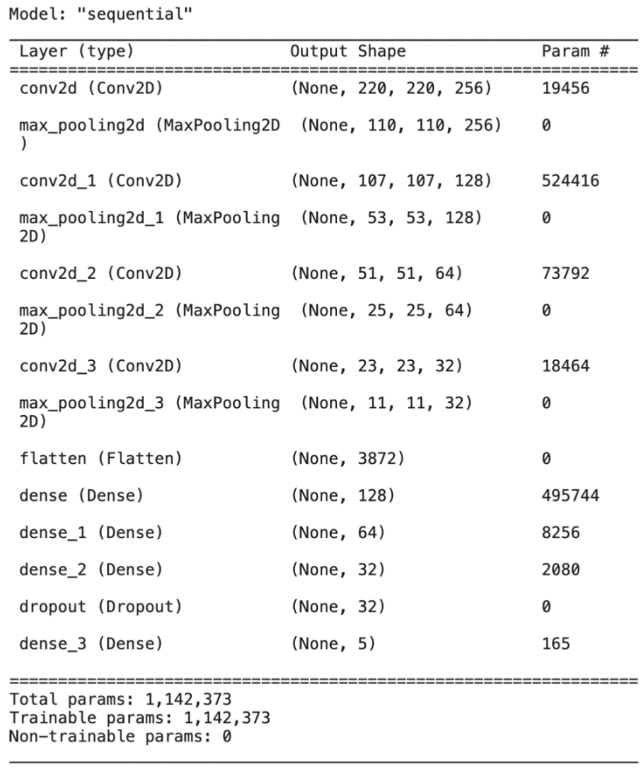
Max Pooling Layers: Four max pooling layers are strategically placed to down sample the information and retain the most crucial aspects.

Dense Layer (Fully Connected Layer): A single dense layer helps in forming connections and making predictions based on the extracted features.

Dropout Layer: To enhance model generalization and mitigate overfitting, we've included a dropout layer in our architecture.

Our training process is optimized for performance and stability. We utilize the Adam optimizer, a popular choice for gradient-based optimization. Additionally, we've implemented a mechanism to dynamically adjust the learning rate during training. If the validation accuracy remains unchanged for two consecutive epochs, the learning rate is reduced by half. This approach helps the model avoid getting stuck in a plateau and promotes continued learning.
An interesting feature of our custom CNN architecture is the diversity in kernel sizes within the convolutional layers. We employ kernel sizes of (3,3), (4,4), and (5,5), allowing the network to extract features of different sizes from the input data.

The input layer of our model is configured with a size of (224,224,3). This signifies that our model can process color images with dimensions of 224 pixels in width, 224 pixels in height, and three channels, which is typical for RGB color images.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 220, 220, 256)     19456

 max_pooling2d (MaxPooling2D  (None, 110, 110, 256)    0
 )

 conv2d_1 (Conv2D)           (None, 107, 107, 128)     524416

 max_pooling2d_1 (MaxPooling  (None, 53, 53, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 51, 51, 64)        73792

 max_pooling2d_2 (MaxPooling  (None, 25, 25, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 23, 23, 32)        18464

 max_pooling2d_3 (MaxPooling  (None, 11, 11, 32)       0
 2D)

 flatten (Flatten)           (None, 3872)              0

 dense (Dense)               (None, 128)               495744

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dropout (Dropout)           (None, 32)                0

 dense_3 (Dense)             (None, 5)                 165

=================================================================
Total params: 1,142,373
Trainable params: 1,142,373
Non-trainable params: 0
_________________________________________________________________
```

Figure 7 – Summary of the Finalized Model

6. Performance Metrics

By the following plot is evident that the training of the model has reached a stable state after the initial iteration. The loss vs. epochs curve clearly illustrates this by showing that the loss has plateaued and reached a constant value.

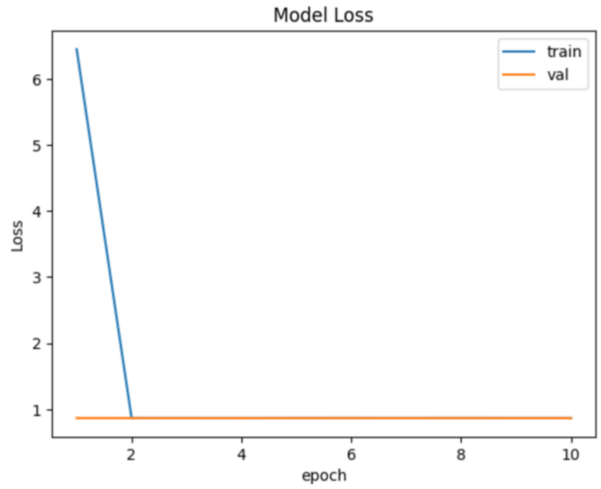


Figure 8 – Plot of Loss vs. epochs

Similarly, the accuracy vs. epochs curve demonstrates that the model's accuracy has also reached a consistent value after the initial training phase.

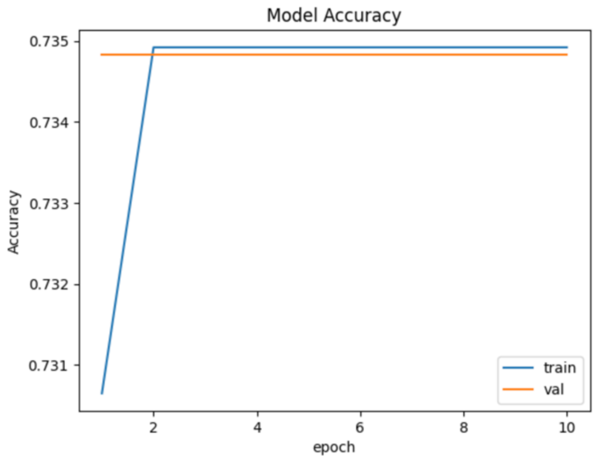


Figure 9 – Plot of Accuracy vs. epochs

After the training phase, the model was evaluated on the tesing dataset consisting of 3511 images of size (224,224) and the following performance metrics were obtained.

Test Loss	-	0.8665
Test Accuracy	-	0.7351

The test loss was measured at 0.8665, indicating a relatively low level of error in the model's predictions. Furthermore, the test accuracy, a key performance metric, was found to be 0.7351, signifying that the model correctly classified approximately 73.51% of the test data. However, this relatively high accuracy is due to the accurate prediction of the majority class.

Additionally, precision, recall, and the F1-Score were calculated to provide a comprehensive understanding of the model's performance.

Precision	-	0.5404
Recall	-	0.7351
F1-Score	-	0.6229

The precision value was determined to be 0.5404, demonstrating the model's ability to accurately identify positive cases. The recall value, which stood at 0.7351, reflects the model's capability to correctly capture a substantial portion of true positive cases. Finally, the F1-Score, a balanced metric that takes both precision and recall into account, was computed to be 0.6229.

7. Challenges

During our project, we encountered several common and challenging issues, with the most prominent being the highly imbalanced nature of the dataset. This imbalance presented numerous obstacles during the training process, as the model tended to predict the majority class when dealing with imbalanced data. To address this, we explored various data sampling and augmentation methods, including oversampling the minority classes and undersampling the majority class to create a more balanced dataset. However, the results were not as anticipated, and accuracy even decreased in some cases.

Another significant challenge was related to feature extraction from the images within each class. The distinguishing features that indicate membership in a particular class were often subtle and challenging to extract. To overcome this challenge, we implemented preprocessing techniques to enhance and emphasize the critical features within the images.

In addition, variations in lighting conditions and image quality posed challenges, as not all images were consistent in these aspects. To mitigate this issue, we utilized a preprocessing function to standardize and equalize the lighting conditions and image quality across the dataset.

Furthermore, image sizes were inconsistent, and many images had higher resolutions than desired. High-resolution images can lead to a large number of input features, which, while potentially informative, can significantly increase the time required to train the model. Therefore, we struggled with resizing and optimizing image sizes to obtain a balance between information content and training efficiency.

8. Future Modifications

An effective improvement to address the issue of data imbalance is to strategically divide the data into multiple stages for classification. Given that one class (in our case, class 0) dominates the dataset, we can start by determining whether a specific data point belongs to this majority class or not. This initial stage focuses on isolating class 0.

Once we have separated the majority class, we can move on to the next stage, which involves identifying the next majority class (in our case, class 2) and deciding whether a data point belongs to that class. We continue this process iteratively, considering each majority class one at a time. In our dataset, classes 1, 3, and 4 have relatively balanced data sets.

As we correctly identify each major class, we gradually transition into a new classification problem with a more balanced data set. For example, after accurately classifying classes 0 and 2, we can proceed to address a classification problem with three classes, each of which now has a more balanced representation.

This staged classification approach simplifies the task and enhances accuracy when dealing with imbalanced data. While it shares similarities with the One-vs-Rest (OvR) method, it differs in that it systematically tackles each majority class, ultimately creating a more balanced and manageable classification problem.

9. References

1. Wilkinson CP, Ferris FL 3rd, Klein RE, Lee PP, Agardh CD, Davis M, Dills D, Kampik A, Pararajasegaram R, Verdaguer JT; Global Diabetic Retinopathy Project Group. Proposed international clinical diabetic retinopathy and diabetic macular edema disease severity scales. Ophthalmology. 2003 Sep;110(9):1677-82. doi: 10.1016/S0161-6420(03)00475-5. PMID: 13129861.
2. https://www.kaggle.com/datasets/tanlikesmath/diabetic-retinopathy-resized
3. Tajudin, Nurul & Kipli, Kuryati & Hamdi, Mahmood & Lim, Lik Thai & D.A.A, Mat & Sapawi, Rohana & Sahari, Siti & Lias, Kasumawati & Jali, Suriati K & Hoque, Mohammed. (2022). Deep learning in the grading of diabetic retinopathy: A review. IET Computer Vision. 16. n/a-n/a. 10.1049/cvi2.12116.
4. https://datagen.tech/guides/computer-vision/vgg16/
5. https://www.kaggle.com/code/rahulrajpandey31/diabetic-retinopathy-1024x1024/notebook