DEEP LEARNING: DIABETIC RETINOPATHY DETECTION USING FUNDUS IMAGE

By

BEH JUN YUE

A REPORT
SUBMITTED TO
UNIVERSITI TUNKU ABDUL RAHMAN
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Ts Dr. Goh Chuan Meng, for providing me with the invaluable opportunity to work on this project with the title of “Deep Learning: Diabetic Retinopathy Detection Using Fundas Image.” His guidance expertise and unwavering support have been instrumental in helping me to take my first step into the field of deep learning and medical image analysis. A million thanks to you for believing in me and guiding me throughout this journey.

To my parents, I owe an immeasurable debt of gratitude for their endless love, encouragement, and sacrifices. Your unwavering belief in me has been always my greatest source of strength, and I am forever thankful for your support.

ABSTRACT

Diabetic retinopathy (DR) is one of the leading causes of blindness worldwide. DR normally come with diabetes when it affects an individual. Since DR is one of the leading causes that cause blindness, early detection of it is much essential to prevent vision loss. However, the traditional method of diagnosing DR is to rely on manual examination of retinal images by ophthalmologists. It is a time-consuming and is highly prone to error process since it is done by using manpower. Nowadays, with the growing number of diabetic patients, there is an urgent need for an efficient and automated solution for DR screening to prevent blindness due to DR, and therefore this project is proposed. This project is aimed to develop a deep learning-based system using Convolutional Neural Networks (CNNs) for the automated detection and classification of DR. The dataset that is applied in this project is obtained from Kaggle "Diabetic Retinopathy 224 x 224 Gaussian-Filtered" dataset, which consist high-resolution fundus images across the five classes: No DR, Mild, Moderate, Severe and Proliferate DR. Due to the significant class imbalance appear in the dataset, some data augmentation techniques such as flipping, rotation, and zooming, along with class weighting are applied to improve the performance. A DenseNet-201 model with transfer learning from ImageNet was employed, enhanced with global average pooling, batch normalization, dropout, and a softmax output layer. Some metrics were used to evaluate the performance of the model such as accuracy, precision, recall, F1-score and confusion matrix analysis. The experiment results show that the proposed DenseNet-201 model achieves strong classification performance and shows promise as a reliable and efficient tool for automated DR screening, supporting early detection and reducing the workload of the ophthalmologists.

Area of Study: Deep Learning, Image Processing

Keywords (Maximum 5): Diabetic retinopathy (DR), DenseNet-201, Data Augmentation, Deep Learning, Medical Images Classification.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DR	Diabetic retinopathy
CNN/CNNs	Convolutional Neural Network(s)
AI	Artificial Intelligence
HIPAA	Health Insurance Portability and Accountability Act
GDPR	General Data Protection Regulation
GAN/GANs	Generative Adversarial Network(s)
OCT	Optical Coherence Tomography
API	Application Programming Interface
JSON	JavaScript Object Notation
CLAHE	Contrast Limited Adaptive Histogram Equalization
PDF	Portable Document Format
RGB	Red-Green-Blue
NDPR	Non-Proliferative Diabetic Retinopathy
GUI	Graphical User Interface
PDF	Proliferative Diabetic Retinopathy

CHAPTER 1

Introduction

This chapter introduces the problem statement and motivation, project scope, project objectives, impact, significance and contribution, background and report organisation.

1.1 Problem Statement and Motivation

Even though CNNs have shown significant help in medical image analysis, there are still some challenges that still limit their full potential in DR detection. One of the major challenges is the limited availability of large, labelled datasets, which is a must for effectively training a deep learning model. This limitation arises due to the sensitivity of the medical data and strict privacy regulations, such as HIPAA in the United States and GDPR in Europe which restrict data sharing and complicate dataset compilation. It is understandable that such regulations are necessary to protect the patient's privacy, but they also created an obstacle when it comes to assembling datasets for model training and testing. Additionally, annotation of medical images requires specialized expertise from trained professionals, particularly ophthalmologists. This will increase both the cost and time that is required to develop a large dataset. Due to the time-consuming nature of labelling medical images across different stages of DR, the availability of comprehensive training data is further limited.

Another challenge is the class imbalance that is commonly found in DR datasets. Most of the retinal images represent early stages such as "No DR" or "Moderate DR", while more advanced stages like "Severe DR" and "Proliferative DR" are underrepresented. This imbalance can cause the model to become biased towards predicting the majority classes, reducing its sensitivity to critical severe cases. A deep learning model that is trained on an imbalanced dataset may achieve a high overall accuracy but will show a lower sensitivity for rare stages, which can delay necessary treatment [1]. It is a must overcome these challenges to build a reliable automated DR screening system.

1.2 Project Scope

The scope of this project is to develop an automated DR screening system by using deep learning technologies. The system is specifically designed for retinal fundus images, focusing on classifying images into five stages of DR. This project does not consider other medical imaging such as brain tumour, optical coherence tomography (OCT) images.

The model should be developed with deep learning with transfer learning to classify fundus images efficiently. Class imbalance should be overcome to make the model perform better. Some preprocessing steps should also be taken before feeding the medical images into the model.

This project should also involves developing a simple desktop application that allows users to upload fundus images and receive automated screening prediction. The dataset used in this project is obtained from Kaggle which is named as "Diabetic Retinopathy 224 x 224 Gaussian Filtered" dataset [2].

1.3 Project Objectives

The objective of this project are as follows:

i. **To develop an automated DR screening system using deep learning techniques.**

   This system is implemented with DenseNet-201 model that is trained with transfer learning. This system can classify retinal fundus images into different DR stages. It can automatically analyse the input fundus images and classify them into five stages of DR without requiring any manual feature extraction.

ii. **To overcome dataset class imbalance through augmentation and class weighting.**

   Data augmentation techniques such as random flipping, rotation, and zooming are applied to expand minority classes, while class weighting adjusts the learning process to ensure fairer treatment of underrepresented DR stages. This improves the model's robustness and detection accuracy across the stages.

iii. **To implement a desktop-based application for user-friendly DR screening.**

The system provides an interface where the users can upload the retinal images or folders and instantly get the DR classification results after they run the detection. The application also supports the features of exporting the results into PDF and Excel formats, making it practical for offline use without requiring any specialized hardware.

1.4 Impact, Significance and Contribution

Diabetic retinopathy is one of the main causes of vision loss globally, especially among individuals with long-term diabetes. Early screening and detection play as an important role in preventing blindness, but many regions still facing shortage of well-trained ophthalmologists and screening infrastructure. This project fulfils the need for scalable and efficient DR screening by development a system that automates the classification of fundas images by using deep learning. The adoption of DenseNet-201 with transfer learning ensures strong classification performance while maintaining efficiency, and the application of data augmentation together with class weighting helps to address the class imbalance challenge that is commonly present in medical datasets.

The key contribution of this project is the development of a practical screening solution that supports early identification of DR stages. User can easily get the classification results by just simple uploading the retinal images that they want to test for to a simple desktop application. This shows that the potential of system to ease the workload of the healthcare professionals and provide a fast preliminary assessment. In addition, the project highlights the effective use of Gaussian-filtered retinal images, transfer learning with DenseNet-201, and augmentation strategies in building a functional DR screening system. Overall, this work contributes a foundation for future research and application in automated medical images classification systems.

1.5 Background

One of the major causes of blindness in the world today is diabetic retinopathy (DR), a serious microvascular consequence of diabetes. Continuous high blood sugar levels will

damage the retinal blood vessels, leading to why diabetic macular degeneration (DR) arises. If it is left untreated, the illness will progress through multiple stages which are: non-proliferative diabetic retinopathy (NPDR), characterized by minor abnormalities in the retina, to proliferative diabetic retinopathy (PDR). There is a greater chance of retinal detachment, macular oedema, and ultimately irreversible blindness when PDR-related abnormal blood vessel formation occurs on the surface of the retina [3][4].

The traditional diagnostic method for DR involved a trained, experienced ophthalmologist, who will manually inspect the retinal fundas images to identify for the early symptoms of the disease. It is a method that is said to be labour-intensive, time-consuming and prone to error. Furthermore, some small change of retinal that occurs in the preliminary stages of DR can be often missed out during manual examination, somehow increasing the risk of delay diagnosis and treatment, which will cause some serious issues. The increasing number of diabetic patients further complicates the problem, straining healthcare systems and highlighting the need for automation in DR diagnosis [3][5].

Nowadays, deep learning, or more accurately, Convolutional Neural Networks (CNNs), has shown great promise in revolutionizing medical images analysis by providing an automated, fast and accurate diagnostic tools. CNNs is a type of artificial intelligence (AI) that is designed to process image data. It has proven its highly effective at automatically extracting and learning the hierarchical features from the retinal images. Unlike the traditional algorithms which are more rely on manual feature extraction, CNNs can performs a better job since it can detect subtle patterns such as microaneurysms, haemorrhages, and exudates which may be an indicative of early-stage DR. These abilities of it significantly reduces the chance of human error and allows for a more consistent diagnoses across large datasets [6].

Severe studies have proved the effectiveness of CNNs in diabetic retinopathy detection. For instance, Gulshan *et al*. developed a deep learning algorithm that surpassed the diagnostic accuracy of most ophthalmologists, which achieved a high sensitivity of 97.5% and specificity of 93.4% in detecting diabetic retinopathy from retinal fundas images [7].

In the real world, there will always be a situation where the dataset class is imbalanced. This happened because early-stage DR cases such as “No DR” and

"Moderate" are far more common than advanced stages like "Severe" or "Proliferate DR". To overcome such issues, this project applied extensive data augmentation techniques such as flipping, rotation, and zooming to generate additional samples for the minority classes. In addition, class weighting was used during training to further balance the learning process and reduce model bias toward the majority classes. These strategies help the DenseNet-201 model to improve sensitivity in detecting rare but critical DR stages. By integrating augmentation and class weighting into the training pipeline, the system achieved more balanced performance across all categories, enhancing the accuracy, efficiency, and reliability of automated DR screening.

1.6 Report Organisation

This report is organized into seven chapters. Chapter 1 introduces the project background, problem statement, objectives, scope, and overall contributions. Chapter 2 focuses on the technologies utilized, reviews related studies by outlining their strengths and limitations and highlights how this project addresses those gaps. Chapter 3 discusses the methodology and the overall flow of the system from data handling to model evaluation. Meanwhile, Chapter 4 presents the system design, including the architectural structure, component interactions, and interface planning. Chapter 5 details the implementation environment and configuration, as well as the key modules that realize the system. Chapter 6 reports the system testing procedures, performance metrics, results, and a discussion of findings and limitations. Finally, Chapter 7 concludes the work and provides recommendations for future improvements and deployment readiness.

CHAPTER 2

Literature Reviews

This chapter elaborates on the technology used and previous works that contribute to the development of the project, highlighting their limitations and strengths.

2.1 Review of Technologies

2.1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are known as one of the most used deep learning models when it comes to completing tasks such as image classification and image analysis. The concept of CNNs is inspired by the human visual cortex where it can extract hierarchical features from the images, ranging from simple edges to some more complex textures and structures. The capability of it to do so allows them to be more suitable for medical imaging tasks where the disease indicators often appear as subtle abnormalities.

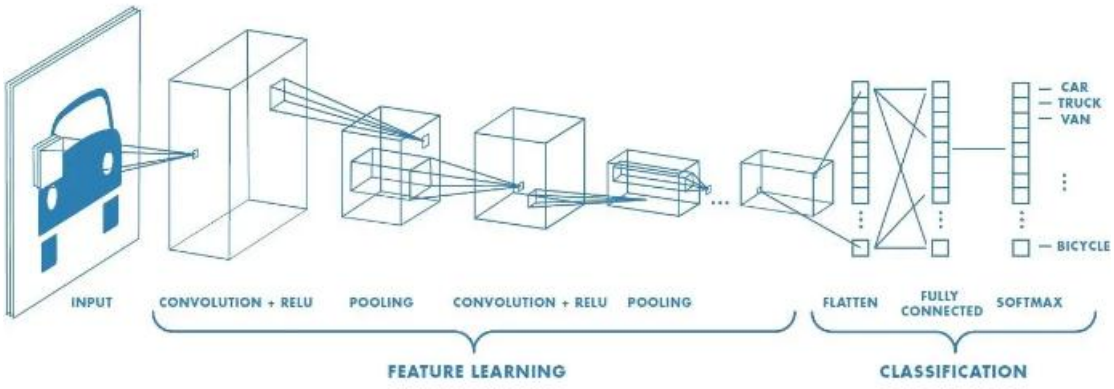


Figure 2.1.1.1 Example on how Convolutional Neural Networks (CNNs) works [8]

When it comes to the task like diabetic retinopathy (DR) detection, CNNs have proven transformative. Some fine-grained retinal features such as microaneurysms, hemorrhages, and exudates can be consistently identified by the CNNs, in which these

features are often difficult to detect through manual examination by ophthalmologists. This has made CNNs to be highly effective for supporting large-scale and automated DR screening programs.

A landmark study that is done by Gulshan *et al.* applied CNNs to a large dataset that consists of 128, 175 retinal fundus images for binary classification of referable versus non-referable DR. Their model achieved a sensitivity of 97.5% and a specificity of 93.4%, surpassing the diagnostic accuracy of most ophthalmologists [7]. This result has strongly proven the potential of CNNs to deliver a scalable, reliable, and clinically relevant results in DR detection.

Meanwhile, there is also a study that is done by Pratt *et.al.* which is focused on multi-class DR classification. They trained a CNN to distinguish between different severity levels from "No DR" to "Proliferate DR". Their model has successfully achieved an accuracy of approximately 75%, showing the feasibility of CNNs for stage-wise classification of DR, but at the same time also highlighting the challenges in achieving a high accuracy across all categories [6].

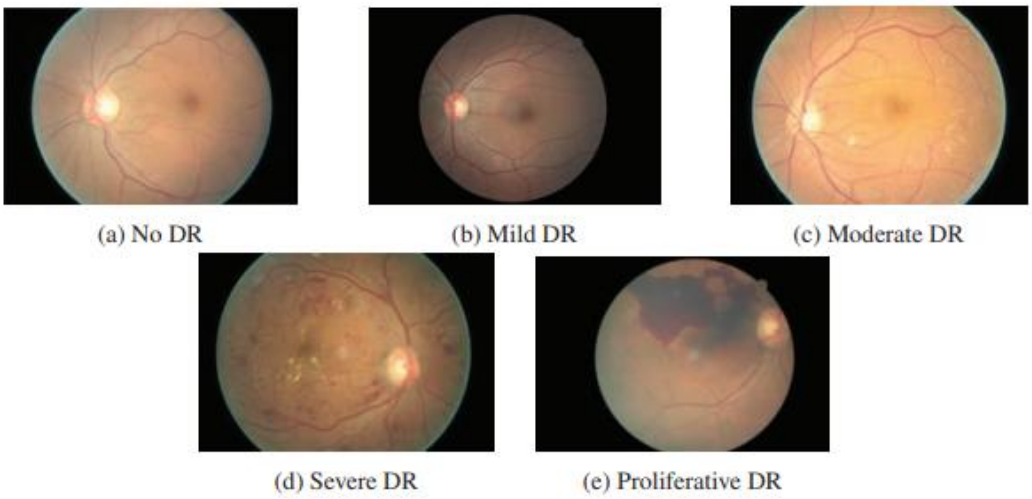


Figure 2.1.1.2 Example of Fundas Images of Diabetic Retinopathy [6]

These studies collectively highlight both the strengths and weaknesses of CNNs in DR detection. On one hand, CNNs provide an automated feature extraction, scalability, and strong diagnostic accuracy. In addition, there are also studies that identified class imbalance as a very common challenge in DR datasets in which the early stages such as "No DR" and "Mild DR" are overrepresented [1]. This imbalance issue always causes the CNN model to perform well on majority classes but less

effectively in rare yet clinically critical cases. These challenges can be overcome by implementing some advanced strategies such as transfer learning, data augmentation, and specialized CNN architecture to improve its reliability in multi-class DR classification.

2.1.2 Transfer Learning

Transfer learning has emerged as a powerful technique in medical image analysis, particularly for deep learning models that typically require large amounts of data to perform effectively. In transfer learning, a model is pre-trained on a large, general-purpose dataset such as ImageNet which contains millions of labeled images. The model is then repurposed for a specific task such as diabetic retinopathy (DR) detection with a much smaller and more specific dataset. The pre-trained models that already possess knowledge about the general features like edges and texture. These can be reused for the new task.

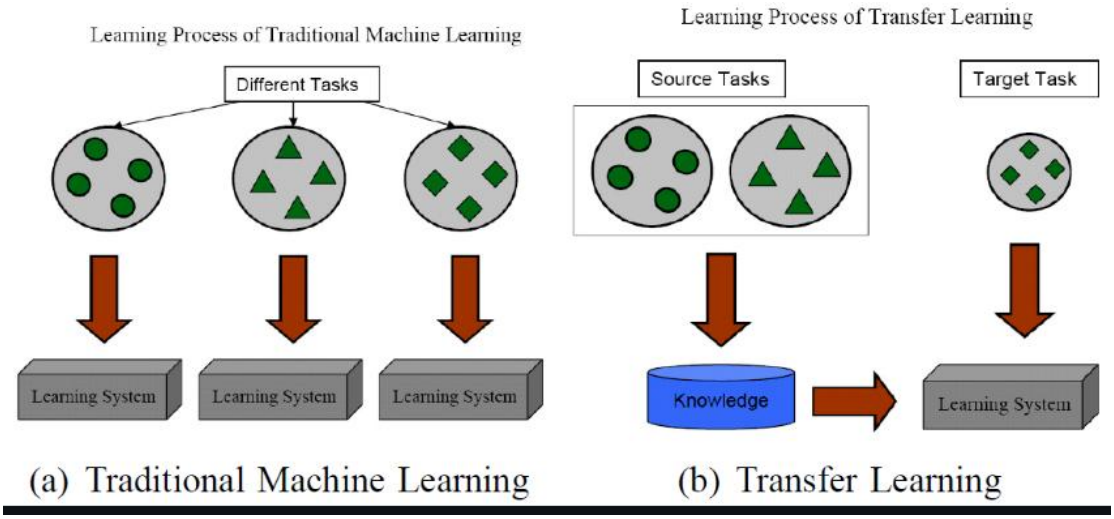


Figure 2.1.2.1 Example on how Transfer Learning works [12]

Shin *et al.* [9] were among the first to systematically discover the first to systematically investigate the effectiveness of transfer learning in medical imaging. Their study examined various CNN architecture and dataset scales, showing that initializing a model with weights pre-trained on ImageNet does significantly improve

the performance in computer-aided detection tasks compared to training them start from scratch. Their findings highlighted that transfer learning not only reduced the training time but also enhanced the accuracy when the labeled medical images dataset were limited, making it a practical solution in clinical applications.

For the specific domain of DR classification, Kandel and Castelli [10] have provided a comprehensive review of studied applying transfer learning to retinal fundus images. They observed that CNN models pre-trained on ImageNet, such as VGG, ResNet, and DenseNet, consistently outperformed models trained from scratch. Their review emphasized that the transfer learning convergence mitigates the effects of limited annotated data and has become a standard practice when it goes to DR image classification. This underscores the adaptability of transfer learning as a method to improve model generalization across diverse DR datasets.

However, there is also study that is done by Raghu *et.al.* [11] critically evaluated transfer learning in medical imaging and proposed that ImageNet pre-training does not always provide a substantial gain, especially when it comes to the target dataset is sufficiently large or when the domain-specific features differ significantly from natural images. Their study found out that in some cases smaller CNNs trained directly on a medical dataset will be performed comparably to the models that are fine-tuned from ImageNet. This has raised the important consideration on when transfer learning is most beneficial.

In conclusion, these literatures reviews about transfer learning indicate that while transfer learning has become a cornerstone in DR detection, its effectiveness is still needed to depend on some factors such as dataset size, quality, and similarity between the source and target domains. The dual perspective of study that is done by these researchers has clearly proven the success of transfer learning in DR studies alongside its potential limitations which highlights the importance of careful evaluation when applying transfer learning in medical image analysis.

2.1.3 Operating System

The operating system that is used for this project is Microsoft Windows 11. An operating system that offers a stable environment for desktop software development.

Windows provides support for Python environment, third-party libraries, and GUI frameworks. It always offers good compatibility with Python, machine learning frameworks, and GUI toolkits [17]. This makes it suitable for both academic and professional use. Windows 11 also provides a very user-friendly interface, stability, and robust resource management, which is a must when it comes to supporting application development and usability in diverse environments.

2.1.4 Python

Python is a high-level, interpreted programming language knowns for its simplicity, readability, and strong community support. It has become one of the famous languages that is used for scientific computing, machine learning, and application prototyping [18]. Python's ecosystem is normally supported by a rich collection of libraries such as TensorFlow and Keras for deep learning, OpenCV for image processing, Pandas for data handling, SciPy for data handling and numerical analysis [19] [20] and ReportLab or OpenXL for report generation. Its design philosophy emphasizes the developer's productivity and the code clarity, which allow a rapid development to carried out without sacrificing any flexibility. This availability of virtual environment also ensures that the reproducibility and isolation of dependencies, which is an essential in the modern software project [18] [20].

2.1.5 Summary of the Technologies Review

The reviewed technologies collectively form the foundation for developing and deploying intelligent desktop applications. Convolutional Neural Networks (CNNs) provide automated feature extraction and strong performance in visual recognition tasks, though they require strategies such as augmentation and specialized architectures to overcome challenges like class imbalance [6], [7]. Transfer learning further enhances CNN effectiveness by reusing pre-trained weights from large datasets such as ImageNet, reducing training costs and improving performance on smaller, domain-specific datasets. However, its success depends on dataset characteristics and similarity between source and target domains, requiring careful evaluation [9]– [11]. Python, as a high-level programming language, plays a central

role due to its readability, strong community support, and rich ecosystem of scientific libraries, including TensorFlow, Keras, OpenCV, Pandas, and SciPy, which facilitate deep learning, image processing, and numerical computation [18]– [20]. Finally, Microsoft Windows 11 provides a stable operating environment with strong support for Python, third-party frameworks, and GUI toolkits, ensuring compatibility and usability across academic and professional contexts [17].

These technologies review can offer a balanced stack of algorithms, programming tools, and computing platforms that is suitable when it comes to developing a scalable, efficient, and user-friendly application.

2.2 Review of Previous Work

2.2.1 Convolutional Neural Networks for Multi-Class Diabetic Retinopathy Detection (Pratt *et al*., 2016)

This paper [6] proposed a deep convolutional neural network (CNN) to classify retinal fundus images into the five standard stages of diabetic retinopathy (No DR, Mild, Moderate, Severe, and Proliferative DR) using the Kaggle EyePACS dataset. Their method began with preprocessing, where images were resized from high resolution (over 6 megapixels) down to 512 x 512 pixels and colour normalization was applied to minimize lighting differences. To increase the robustness, the paper applied real-time data augmentation during training, such as random rotation, flips and shifts, so that the model could better generalize across varied imaging conditions.

The CNN architecture that is proposed in this paper followed a typical layered approach, with convolutional layers to extract hierarchical features such as textures and lesion shape. Batch normalization is used to stabilize the learning and max pooling is applied to progressively reduce spatial dimensions. The final dense layers of the CNN architecture are with softmax activation produced probability for each stage of the five DR stages. The paper also introduced class weighting in the loss function to overcome the class imbalance problem (with advanced stages being much less common), this successfully make the minority classes in the dataset to have a more stronger impact during the training session. Some other measures such as dropout or L2 regularization

were also applied to overcome overfitting problem, while the model was optimized by using stochastic gradient descent with Nesterov momentum.

Fig 2: Network architecture

Figure 2.2.1.1 Network architecture for Pratt *et al.* paper [6]

This design allowed the network to automatically detect important retinal features such as microaneurysms, haemorrhages, and exudates, which are the subtle indicators of disease progression and are often difficult to capture with traditional hand-crafted feature methods that is done by ophthalmologists. However, even though this paper had achieved strong specificity (95%) and an overall accuracy of ~75% on validation data, the sensitivity was relatively low which only achieved a sensitivity of 30%. This clearly show that the model is struggling with correctly detecting the minority-class images, particularly in the severe and proliferate categories.

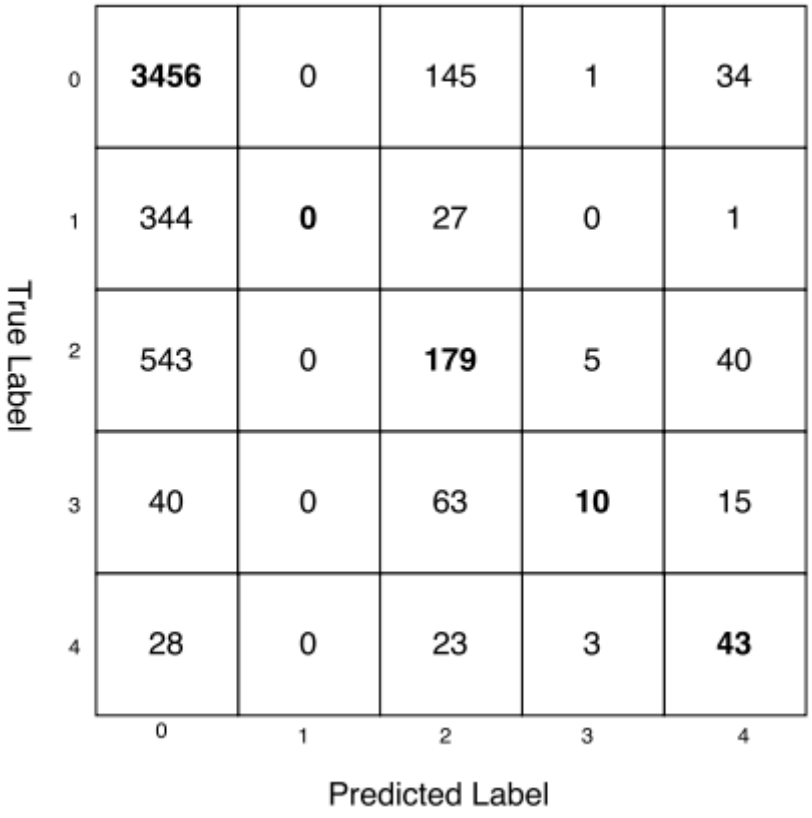


Figure 2.2.1.2 Confusion matrix of Pratt *et al.* paper [6]

2.2.2 Custom Convolutional Neural Network for Diabetic Retinopathy Detection on Gaussian-Filtered Images (Rashika *et al.*, 2019)

This paper [13] proposed a custom convolutional neural network (CNN) for the multi-class classification of diabetic retinopathy using the "Diabetic Retinopathy 224 x 224 Gaussian-Filtered" dataset that is obtained from Kaggle website. The preprocessing step involved applying a Gaussian filter to the fundus images in order to reduce high-frequency noise while preserving key retinal structures such as blood vessels and the optic disc boundary. This smoothing process helped emphasize global retinal features and make it easier for the CNN to focus on the clinically relevant lesions such as microaneurysms, hemorrhages, and exudates. These subtle retinal abnormalities are critical for distinguishing among the five stages of DR which are: No DR, Mild, Moderate, Severe and Proliferative DR.

For the architecture, the author described the use of a custom CNN configured with convolutional, pooling, and fully connected layers, where the ReLU was applied as the activation function in the intermediate layers and softmax was used at the output layer to generate class probabilities.

The experimental results showed that the model achieved an overall training accuracy of 84.84% in the task of classifying five class of DR class. The training and validation accuracy/loss curves were presented to illustrate the learning progress, but no independent test-set performance was explicitly reported. This was identified as a limitation, since without external validation, it is unclear whether the model would generalize well to unseen clinical data. Additionally, the study emphasized that the relatively shallow architecture of the custom CNN, compared to more advanced models like DenseNet or ResNet, limited its feature extraction capacity. On the other hand, a notable strength highlighted by the authors was the simplicity of the model design, which made it computationally less expensive and easier to train, thus making it suitable for use in environments with limited hardware resources.

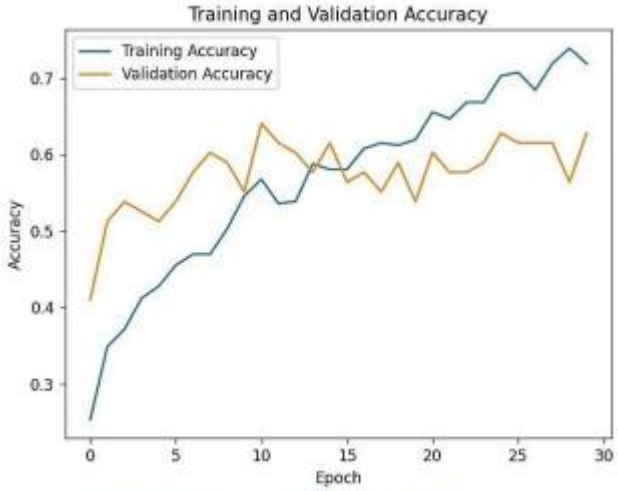


Fig 3. Training and Validation Accuracy

Figure 2.2.2.1 Training and Validation Accuracy for Rath's paper [13]

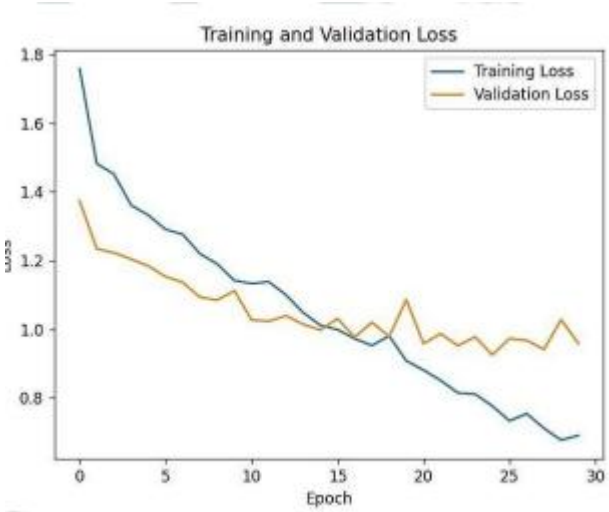


Fig 4. Training and Validation Loss

Figure 2.2.2.2 Training and Validation Loss for Rath's paper [13]

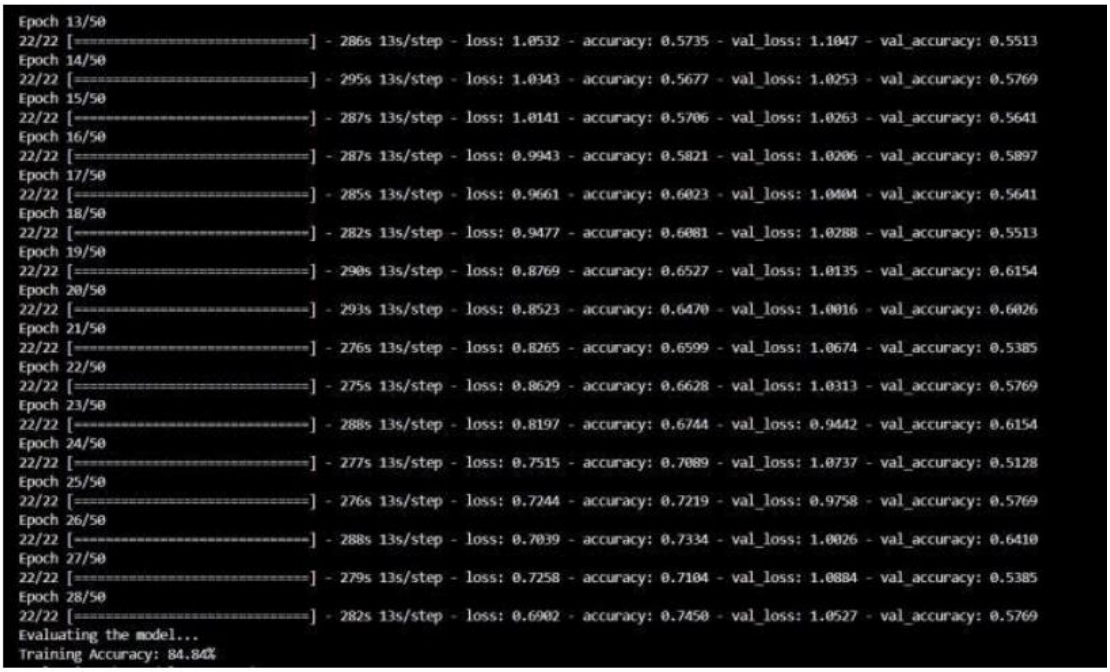


Fig 5: Training Accuracy of 84.84%

Figure 2.2.2.3 Training Accuracy of 84.84% for Rath's paper [13]

2.2.3 Convolutional Neural Network on Gaussian-Filtered Retinal Images (Ezekiel *et al.*, 2020)

Ezekiel, Taylor, and Deedam-Okuchaba (2020) [14] investigated a CNN-based, five-class DR grader trained on Kaggle "Gaussian-filtered" fundus image set. The dataset

that is used in paper was separated into the different severity levels of DR stages such as No DR, Mild, Moderate, Severe, Proliferative. The images in the dataset were pre-processed by applying Gaussian filtering to reduce noise and enhance important retinal structures, such as blood vessels and lesion boundaries.

The authors implemented a custom convolutional neural network built with multiple convolution and pooling layers to progressively extract image features, followed by fully connected layers to perform classification across the five classes. The design allowed the model to capture both low-level features such as vessel contours and higher-level patterns including microaneurysms and haemorrhages.

In terms of performance, the network achieved a training accuracy of 81.35% by the eighth epoch. While the study presented training and validation curves, no independent test accuracy was provided. This was noted as a limitation, as it prevented assessment of the model's robustness on unseen data. Furthermore, the authors emphasized that the model was trained on a relatively small subset of the Kaggle dataset, which may restrict generalizability. However, the deployment of the trained model through a simple web-based interface was highlighted as a strength, since it demonstrated how the CNN could be integrated into a lightweight screening tool for automated predictions.

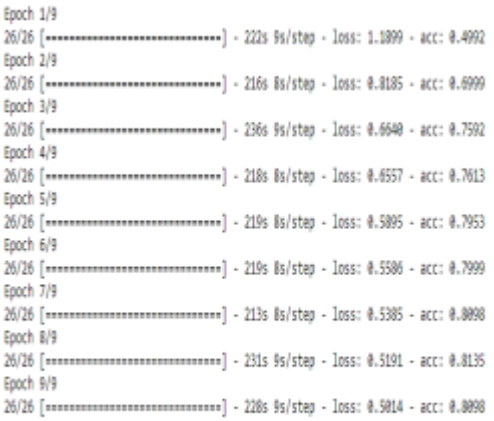


Figure 7: showing training epochs of the model with a loss and accuracy values of the model

Figure 2.2.3.1 Training epoch for Ezekiel *et al.* paper [14]

Figure 8: showing a graphical representation of training accuracy against training epochs

Figure 2.2.3.2 Training accuracy for Ezekiel *et al.* paper [14]

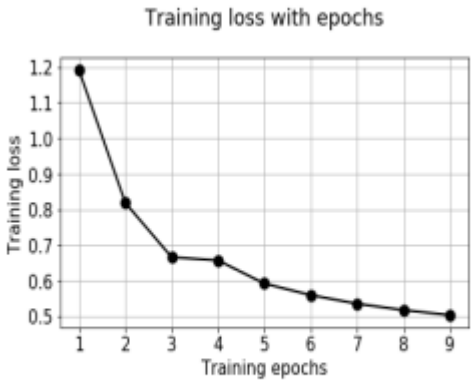


Figure 9: showing a graphical representation of loss values against training epochs

Figure 2.2.3.3 Training loss for Ezekiel *et al.* paper [14]

2.2.4 ResNet34 for Diabetic Retinopathy Detection on Gaussian-Filtered Fundus Images (Verma *et al.*, 2020)

Verma, Pradhan, Rath and Agrawal (2020) investigated the application of deep learning for multi-class DR detection by using the Kaggle "Diabetic Retinopathy 224 x 224" Gaussian- Filtered dataset [15]. Notably, Sovit Ranjan Rath, who originally prepared and uploaded this Gaussian-filtered dataset to Kaggle was also one of the co-authors of the paper, directly linking the study to the dataset's creation. The preprocessing steps employed Gaussian filtering to smooth the image noise and enhance the global retinal structures, with the goal of highlighting pathological features while reducing the

variability that us cause by uneven lighting. The fundus images that are obtained from the Kaggle dataset was separated into the five stages of DR which are: No DR, Mild, Moderate, Sever, and Proliferate DR.

This paper implements a ResNet-34 architecture which is a residual neural network that is composed of 34 layers. The defining feature of ResNet is skip connection, which enable the model to maintain gradient flow across layers, thereby avoiding the vanishing gradient problem that often occurs in deep CNNs. In this context, the ResNet-34 model was tasked with extracting hierarchical lesion feature from the Gaussian-pre-processed images. Early layers focused on fundamental structures such as vessel patterns, while deeper residual blocks learned more complex lesion indicators such as haemorrhages and exudates.

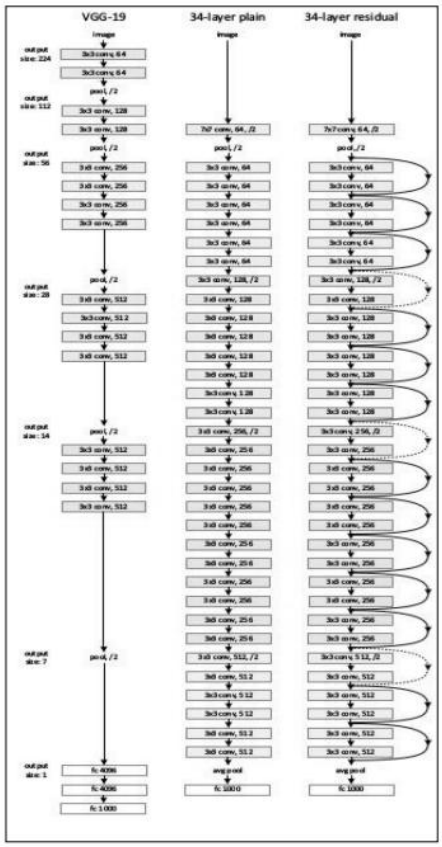


***Figure 3**. ResNet34 blocks*

Figure 2.2.4.1 ResNet-34 for Verma *et al.* paper [15]

For the Gaussian-filtered images, the reported results were not that good when it comes to compared with those achieved with coloured or grayscale inputs. The best training loss was 0.478799, with a validation loss of 0.578836 and an error rate of

0.221311, corresponding to a validation accuracy of approximately 77.9%. Confusion matrix analyses further showed the frequent misclassification among minority classes, particularly in the advanced stages of diabetic retinopathy. The confusion matrix clearly show that those minority classes is having a very small portion of testing and having a low per class accuracy which show that the class imbalance does not been overcome by this paper. The authors concluded that while Gaussian filtering successfully reduced noise, it also removed fine-grained lesion details essential for accurate classification, which contributed to the weaker results observed on this dataset variant.

epoch	train_loss	valid_loss	error_rate
18	0.456303	0.551894	0.207650
19	0.483539	0.548007	0.214481
20	0.441782	0.541847	0.211749

***Table 1**. Last three epoch results for colored retina images*

epoch	train_loss	valid_loss	error_rate
18	0.492623	0.516096	0.200820
19	0.512907	0.520497	0.200820
20	0.480602	0.521209	0.196721

***Table 2**. Last three epoch results for grayscale retina images*

epoch	train_loss	valid_loss	error_rate
17	0.519309	0.587234	0.229508
18	0.510745	0.579334	0.224044
19	0.478799	0.578836	0.221311

***Table 3**. Last three epoch results for Gaussian filtered retina images*

Figure 2.2.4.2 Results for Verma *et al.* paper [15]

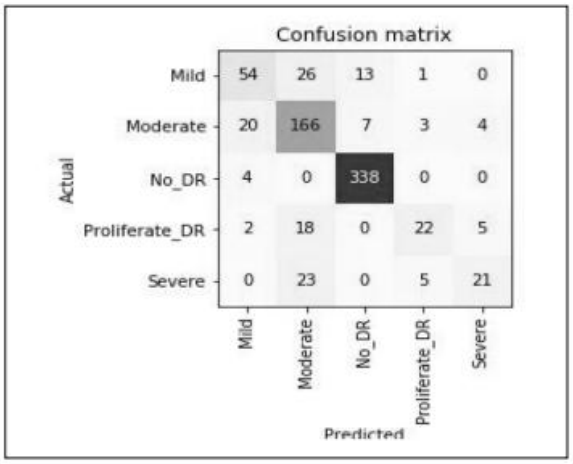


***Figure 7**. Confusion Matrix for Colored Images Validation*

Figure 2.2.4.3 Confusion matrix for Verma *et al.* paper (1) [15]

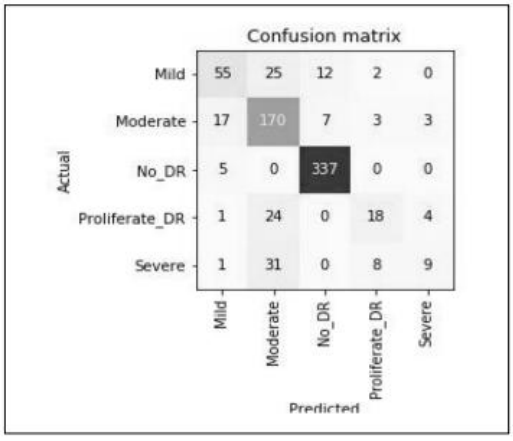


***Figure 8**. Confusion Matrix for Grayscale Images Validation*

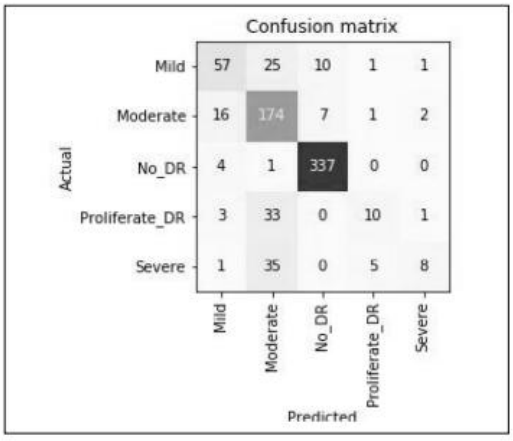


***Figure 9**. Confusion Matrix for Gaussian Filtered Images Validation*

Figure 2.2.4.4 Confusion matrix for Verma *et al.* paper (2) [15]

2.2.5 DenseNet-201 for Multi-Class DR on Gaussian-Filtered Fundus Images (Jeganathan, 2024)

Jeganathan (2024) trained a deep learning system for classifying five class of DR stages by using the Kaggle Gaussian-filtered retinal fundus dataset curated by Sovit Ranja Rath [16]. The pipeline of this paper begins by decoding the PNG fundus photos to floating-point tensors and organizing them into the standard DR categories (No DR, Mild, Moderate, Severe, Proliferative). This paper overcomes the dataset's class imbalance balance by study the augmented minority classes (Severe, Mild, Moderate, Proliferate) with additional transformed images such as random flips, rotations, and zooms. Then, the paper combined both the original and augmented samples into a single training store for efficient loading. Although the text briefly mentions image dimensions of "~224×224," the implementation uses a DenseNet-compatible input resolution and stratified splitting to form the training/validation sets. These steps were intended to suppress acquisition noise through Gaussian prefiltering, standardize the inputs, and expose the model to lesion variability while preserving diagnostically relevant detail such as microaneurysms, haemorrhages, and vessel changes.

For the model of this paper, it adopts DenseNet-201 by using transfer learning with ImageNet weights but fine-tuned the entire backbone together with the new classification head and produces a five-way probability output corresponding to the DR stages. DenseNet's dense connectivity facilitates feature reuse and stable gradient flow, which is good for capturing subtle and multi-scale retinal cues in fundus images while keeping the classifier lightweight. This architecture is proved to be a strength of the study since it does help to mitigate vanishing gradients and allowed the network to learn richer lesion features across the depths, outperforming simpler CNN-based methods reported in related works. Training incorporated heavy on-the-fly augmentation consistent with the preprocessing strategy and the history of loss and accuracy was tracked to visualize convergence.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet201 (Functional)     (None, 7, 7, 1920)        18321984
_________________________________________________________________
global_average_pooling2d (Gl (None, 1920)              0
_________________________________________________________________
dropout (Dropout)            (None, 1920)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 1920)              7680
_________________________________________________________________
dense (Dense)                (None, 5)                 9605
=================================================================
Total params: 18,339,269
Trainable params: 18,106,373
Non-trainable params: 232,896
_________________________________________________________________
```

Figure 2.2.5.1 Model used by Jeganathan's paper [16]

When it comes to evaluation, the paper randomly draws 500 images from the same labelled dataset used for training and validation and reported a final accuracy of 94% on this subset, accompanied with a confusion-matrix visualization and scripts to compute precision and F1-score. Although this strong performance highlighted the potential of DenseNet-201 with Gaussian-filtered images, the evaluation was limited to an internal sample rather than an independent test set, and no external dataset validation was conducted. This raised concerns about overfitting and limited generalizability to real-world screening environments. The workflow also saved the model weights and training history to support incremental retraining and reuse, but the study's reliance on internal validation leaves uncertainty regarding robustness across diverse populations or imaging conditions.

```
Epoch 1/20
220/220 [==============================] - 140s 380ms/step - loss: 0.5702 - accuracy: 0.5993 - val_loss: 0.4207 - val_accuracy: 0.6709
Epoch 2/20
220/220 [==============================] - 71s 319ms/step - loss: 0.2968 - accuracy: 0.7687 - val_loss: 0.2994 - val_accuracy: 0.7082
Epoch 3/20
220/220 [==============================] - 72s 327ms/step - loss: 0.1756 - accuracy: 0.8537 - val_loss: 0.2398 - val_accuracy: 0.7455
Epoch 4/20
220/220 [==============================] - 73s 330ms/step - loss: 0.1178 - accuracy: 0.9098 - val_loss: 0.2457 - val_accuracy: 0.7752
Epoch 5/20
220/220 [==============================] - 73s 330ms/step - loss: 0.0843 - accuracy: 0.9392 - val_loss: 0.2330 - val_accuracy: 0.7913
Epoch 6/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0674 - accuracy: 0.9578 - val_loss: 0.3022 - val_accuracy: 0.7642
Epoch 7/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0534 - accuracy: 0.9674 - val_loss: 0.2828 - val_accuracy: 0.8041
Epoch 8/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0502 - accuracy: 0.9669 - val_loss: 0.2866 - val_accuracy: 0.7769
Epoch 9/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0353 - accuracy: 0.9774 - val_loss: 0.2641 - val_accuracy: 0.7990
Epoch 10/20
220/220 [==============================] - 73s 330ms/step - loss: 0.0263 - accuracy: 0.9837 - val_loss: 0.2903 - val_accuracy: 0.7913
Epoch 11/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0325 - accuracy: 0.9785 - val_loss: 0.3599 - val_accuracy: 0.7795
Epoch 12/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0353 - accuracy: 0.9708 - val_loss: 0.2841 - val_accuracy: 0.8159
Epoch 13/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0213 - accuracy: 0.9881 - val_loss: 0.3362 - val_accuracy: 0.7812
Epoch 14/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0252 - accuracy: 0.9846 - val_loss: 0.3276 - val_accuracy: 0.7727
Epoch 15/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0204 - accuracy: 0.9860 - val_loss: 0.3056 - val_accuracy: 0.7990
Epoch 16/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0275 - accuracy: 0.9780 - val_loss: 0.3255 - val_accuracy: 0.7710
Epoch 17/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0247 - accuracy: 0.9829 - val_loss: 0.4073 - val_accuracy: 0.7625
Epoch 18/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0239 - accuracy: 0.9850 - val_loss: 0.2917 - val_accuracy: 0.8092
Epoch 19/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0168 - accuracy: 0.9878 - val_loss: 0.3553 - val_accuracy: 0.7634
Epoch 20/20
220/220 [==============================] - 73s 329ms/step - loss: 0.0220 - accuracy: 0.9867 - val_loss: 0.3401 - val_accuracy: 0.8007
```

Figure 2.2.5.2 Training epoch for Jeganathan's paper [16]

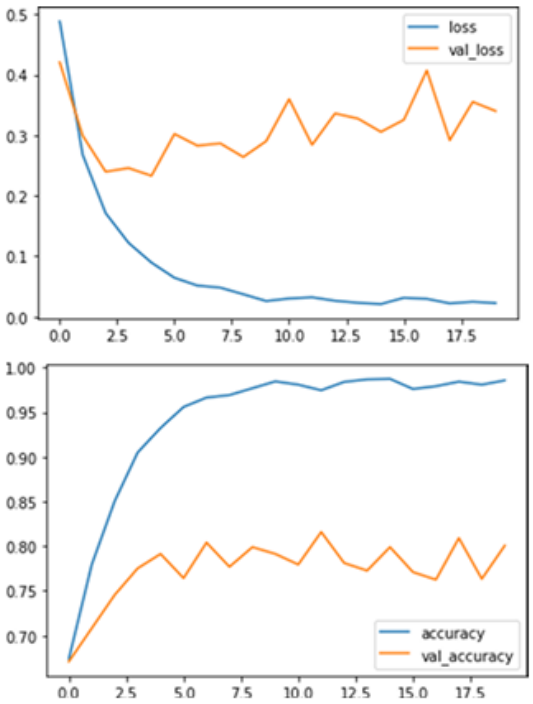


Figure 2.2.5.3 Results of Jeganathan's paper (1) [16]

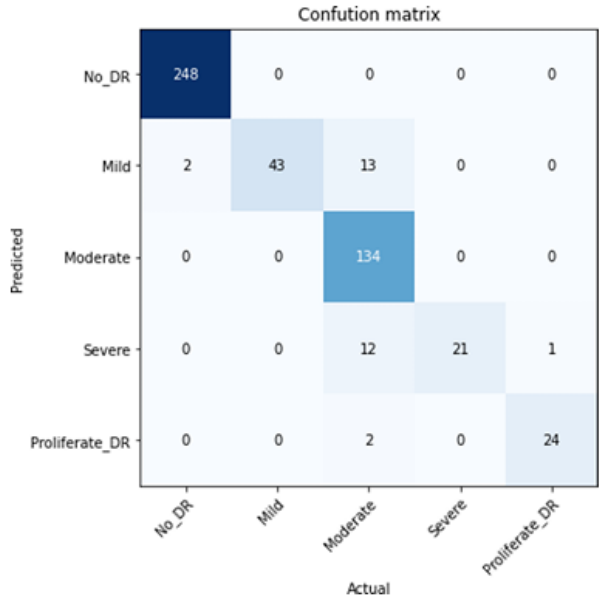


Figure 2.2.5.4 Results of Jeganathan's paper (2) [16]

```
Acuracy Score     : 0.94
Precision Score   : 0.9490079006211178
F1 Socre          : 0.9371199449947756
```

Figure 2.2.5.5 Results of Jeganathan's paper (3) [16]

2.3 Summary of Previous Works

Table 2.3.1 Summary of the previous work

Author(s), Year	Strengths	Limitations	Results
Pratt et al., 2016 [6]	Robust preprocessing and augmentation; showed feasibility of CNNs in large-scale DR detection	Poor sensitivity to minority classes (Severe/Proliferative); struggled with class imbalance	~75% validation accuracy, 95% specificity, 30% sensitivity

Rashika et al., 2019 [13]	Simplicity of architecture; computationally light; effective preprocessing with Gaussian filter	No independent test set; shallow architecture limited feature extraction	84.84% training accuracy
Ezekiel et al., 2020 [14]	Demonstrated real-world applicability with web deployment; captured both low- and high-level retinal features	No independent test accuracy; trained on relatively small dataset subset	81.35% training accuracy (8th epoch)
Verma et al., 2020 [15]	Deeper architecture improved feature extraction; addressed vanishing gradients	Gaussian filtering removed fine lesion details; frequent misclassification in minority classes	Validation accuracy ≈ 77.9% (val. loss 0.5788, error rate 0.2213)
Jeganathan, 2024 [16]	Dense connectivity captured multi-scale features; superior performance; data augmentation improved balance	Evaluation used 500 images from the same dataset as training/validation, so the reported 94% accuracy may not generalize well.	94% accuracy on 500-image holdout set

……

CHAPTER 3

System Method/Approach

This chapter will discuss about the methodology for each trained DenseNet-201 model and for system application and system specifications. Besides, a timeline of the project will also be provided.

3.1 Design Specifications

3.1.1 Methodology of Trained DenseNet-201 Model

The aim of this project is to develop an automated diabetic retinopathy detection system by using deep learning techniques. The methodology in this project involves two main phases which are data augmentation to balance the retinal fundus image dataset, and fine-tuning a DenseNet-201 CNN model to classify the fundus images into the five stages of DR.

This project adopts the overall method that is proposed by Jeganathan [16], who trained a DenseNet-201 model on the Kaggle "Diabetic Retinopathy 224 x 224 Gaussian-Filtered" dataset [2]. In their work, the pipeline consisted of image decoding, resizing to 224 x 224 normalization, generic augmentation, DenseNet-201 fine-tuning, and final evaluation using the accuracy and confusion matrices. The study that is done by Jeganathan demonstrates that DenseNet-201 combined with augmentation achieved a strong internal validation accuracy of 94%.

For this project, the same DenseNet-201 backbone with ImageNet pre-trained weights was adopted. The dataset was pre-processed by decoding all the PNG images that is obtained from the dataset [2]. Then, resizing into 224 x 224, and normalizing them to the range [0,1]. Unlike the article's method where it does a single 75/25 split with a 500-image random evaluation subset. This project, meanwhile, enforces a stratified 70/20/10 split into the training, validation, and test sets to ensure that a fair class representation and a clean held-out evaluation set is present.

To overcome the class imbalance issues, this project diverges from the paper that is done by Jeganathan [16]. Instead of augmenting all the minority class by the same number of images, a targeted augmentation counts were applied, which is 700 for

mild, 300 to moderate, 900 to severe and 800 for Proliferative DR samples were generated using horizontal flipping and random rotation. Besides, the class weight that is mentioned at paper [6] also been computed dynamically and applied during training, ensuring that minority classes contributed proportionally to the optimization. These steps directly respond to the dataset's skewed distribution and aim to improve classification robustness for underrepresented DR stages.

For model training, the DenseNet-201 backbone was fine-tuned end-to-end, followed by a global average pooling layer, drop out (0.5), batch normalization, and a softmax dense layer with five outputs. The model was compiled with the Adam optimizer with a learning rate of 1e-4 and binary entropy loss. It is also trained with a batch size of 16 for 20 epoch. A ModelCheckpoint callback was used to save the best model based on the validation accuracy meanwhile the complete model at the last epoch was also preserved.

For evaluation, this project goes beyond the article's simple accuracy report. The performance was measured on the held-out test set using accuracy, precision, F1-score and normalized confusion matrices. Per-class accuracy was also exported to JSON for detailed analysis of model behaviour across the DR stages.

Finally, the model is then export for deployment purposes. In addition to save the training weights, the complete trained models were exported in both keras and TensorFlow SavedModel formats, along with the class label metadata. This design makes the model directly portable into the project's desktop-based screening application, enabling the real users to upload retinal fundus images and receive automated classification results.

The novelty of this work is lies in its stratified dataset split, targeted augmentation with variable counts, application of class weights, validation-based checkpointing, and deployment-ready exports, all of which extend from the original methodology of Jeganathan [16] to create a more robust and practical screening system.

3.1.2 Methodology of the System Application

The methodology that is applied to the desktop application is more focuses on integrating the trained DenseNet-201 in section 3.1.1 into a more user-friendly platform that allows automated diabetic retinopathy (DR) detection without requiring the end-user to have a technical knowledge about it. The development of the system will follow

a structured pipeline which consists of image input, preprocessing, inference, and output reporting, each of which ensures that the model operate reliably in real-world usage.

The process begins with the image input stage, where the users will interact with a PySide6-based graphical user interface (GUI). The interface provides a function for uploading either a single image or a batch of retinal fundus images in common formats such as JPEG or PNG. Once uploaded, the selected images are then displayed in a queue within the interface, which allow the user to make sure on the files before they start to run the detection for analysis.

After the submission, the images will then proceed to the pre-processing stage, in which it will ensures that all the input data will be conformed to the same standard as the training dataset. Each image is automatically resized to 224 x 224 pixels to match the input requirement of DenseNet-201. A fundus cropping procedure is applied to isolate the retinal region and remove unnecessary background, while CLAHE which stands for Contrast Limited Adaptive Histogram Equalization) will be used to improve the contrast of the images and will highlight the subtle retinal features. Finally, normalization is applied by using DenseNet's preprocessing function so that the pixel values are scaled consistently with the model's training pipeline. These steps collectively guarantee that every input image is standardized before classification.

The inference stage follows, where the exported DenseNet-201 model in keras format is loaded into the memory at the application startup. When a pre-processed image is passed to the model, it will generate a probability distribution across the five DR classes which are No DR, Mild, Moderate, Severe, and Proliferate DR. the class with the highest probability is then selected as the final prediction, and the corresponding class accuracy is recorded. This allows the application to deliver robust and consistent classification in real time, even when processing multiple images sequentially.

Finally, it is the final stages which is output and reporting, where the results are presented in two complementary ways. First, the predictions are displayed directly within the GUI, showing the filename, predicted label, per-class test accuracy values derived from the prior evaluations. This provides immediate feedback to the user. Second, the application offers a reporting feature to support documentation and record-keeping. The users can export the results into a PDF report, which includes the project

logo, image thumbnail, predicted labels, and associated accuracy. Alternatively, the results can be exported into an Excel file that consists of structured data labels, summary statistics, and shorts of class distributions. These reporting function ensure that outputs are not only viewable in real time but also stored in professional formats suitable for academic, clinical, or administrative purposes.

In shorts, the application methodology transforms the deep learning model into a practical tool for DR screening by providing a structured workflow from image upload to professional reporting. Through automated pre-processing, integrated model inference, and comprehensive result presentation, the desktop application delivers a complete and accessible system for diabetic retinopathy detection.

3.2 System Specification

3.2.1 Hardware

The development laptop used in this project is an Asus Vivobook X513EP_A513EP, equipped with an Intel Core i5-1135G7 processor, an NVIDIA GeForce MX330 graphics card, 20GB of RAM, and a 475GB SSD, running on Windows 11 Home Single Language. This laptop was primarily utilized for system testing, desktop application development using PySide6, and report generation with ReportLab. In contrast, the DenseNet-201 model training and fine-tuning were conducted on Kaggle's cloud-based GPU environment, which can provide the necessary computational power for deep learning experiments and faster execution.

Table 3.2.1.1 Specifications of laptop

Description	Specifications
Model	Asus Vivobook X513EP_A513EP
Processor	Intel Core i5-1135G7
Operating System	Windows 11 Home Single Language
Graphic	NVIDIA GeForce MX330
Memory	20GB RAM
Storage	475GB SSD

3.2.2 Software

The development of this project also involves some software tools and libraries. The table below shows the main software and platforms that is used during the system development process.

Table 3.2.2.1 Software Specifications

Software	Description
Kaggle Notebook Environment	Cloud-based GPU environment used for DenseNet-201 model training, evaluation, and dataset augmentation.
Python	Core programming language for model development, dataset handling, and application coding.
TensorFlow	Deep learning framework for loading and running the trained DenseNet-201 keras model in the desktop app
Keras (within TensorFlow)	High-level API for model construction, export, and inference.
PySide6 (Qt for Python)	Framework for building the final desktop GUI, including batch image upload, progress bar, results table, and report export features
Visual Studio Code (VS Code)	Integrated Development Environment (IDE) used for coding, debugging, and project organization throughout development.
OpenCV & scikit-image	Image preprocessing in the app (fundus crop, CLAHE, resize to 224×224).
NumPy	Array/tensor handling for preprocessing and inference.
ReportLab	Generates PDF reports (logo, thumbnails, labels, accuracy).

openpyxl	Generates Excel reports with auto sizing and charts.
PyYAML	Loads app configuration from app_config.yaml.

3.3 Timelines

3.3.1 FYP1 Timelines

Figure 3.3.1 shows the Gantt chart of the current project. Each deliverable is done as below:

- Week 1 - Dataset preparation and project setup
- Week 2 - Implement GAN model for synthetic image generation
- Week 3 - Train and save synthetic images using GAN
- Week 4 - Implement Filter model
- Week 5 - Set up MobileNetV2 structure
- Week 6 - Train and fine-tune MobileNetV2 model
- Week 7 - Evaluate training performance and adjust parameters
- Week 8 - Integrate trained model into Flask web application
- Week 9 - Implement image upload, prediction and result display
- Week 10 - Conduct system testing and debugging
- Week 11 - Finalize model, clean up outputs, and prepare demo
- Week 12 - Complete report writing and final project submission

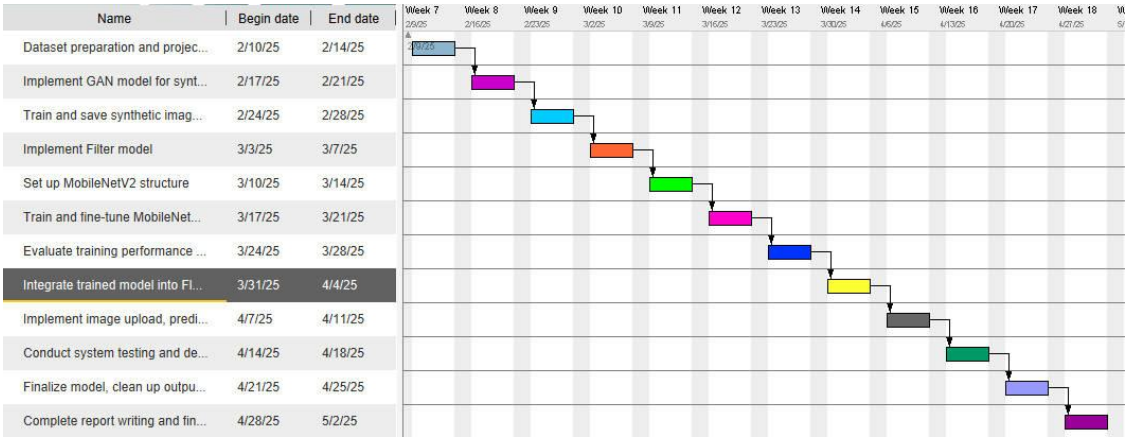
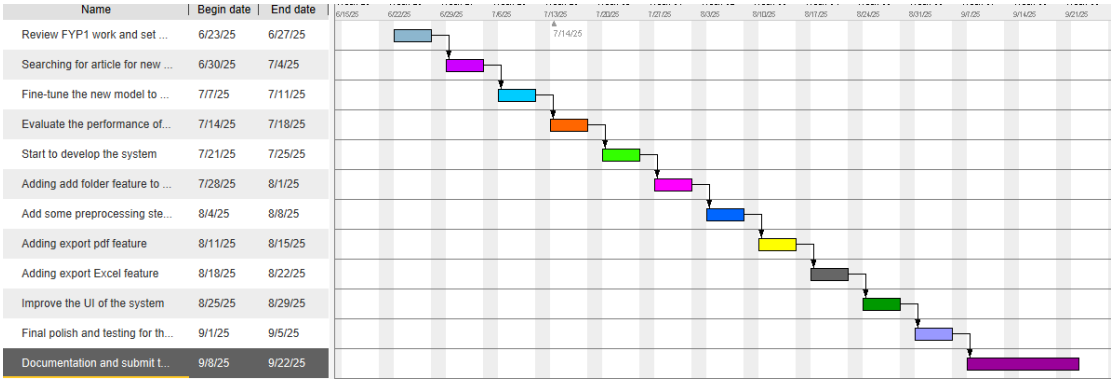


Figure 3.3.1 Gantt Chart for FYP1

3.3.2 FYP2 Timelines

Figure 3.2.2 shows the Gantt chart of the future project. Each deliverable will be done as below

- Week 1 - Review FYP1 work and set project goals
- Week 2 - Searching for article for new model
- Week 3 - Fine-tune the new model to perform a better performance
- Week 4 - Evaluate the performance of the model
- Week 5 - Start to develop the system
- Week 6 - Adding add folder feature to the app
- Week 7 - Add some preprocessing step and error handling
- Week 8 - Adding export pdf feature
- Week 9 - Adding export Excel feature
- Week 10 - Improve the UI of the system
- Week 11 - Final polish and testing for the system
- Week 12 - 13 - Documentation and submit the report



Name	Begin date	End date
Review FYP1 work and set ...	6/23/25	6/27/25
Searching for article for new ...	6/30/25	7/4/25
Fine-tune the new model to ...	7/7/25	7/11/25
Evaluate the performance of...	7/14/25	7/18/25
Start to develop the system	7/21/25	7/25/25
Adding add folder feature to ...	7/28/25	8/1/25
Add some preprocessing ste...	8/4/25	8/8/25
Adding export pdf feature	8/11/25	8/15/25
Adding export Excel feature	8/18/25	8/22/25
Improve the UI of the system	8/25/25	8/29/25
Final polish and testing for th...	9/1/25	9/5/25
Documentation and submit t...	9/8/25	9/22/25

Figure 3.3.2 Gantt Chart for FYP2

CHAPTER 4

System Design

This chapter will show system design diagram such as system architecture diagram, use case diagram and activity diagram. This chapter also included a detail explanation on the development of the system.

4.1 System Design Diagram

4.1.1 System Architecture Diagram

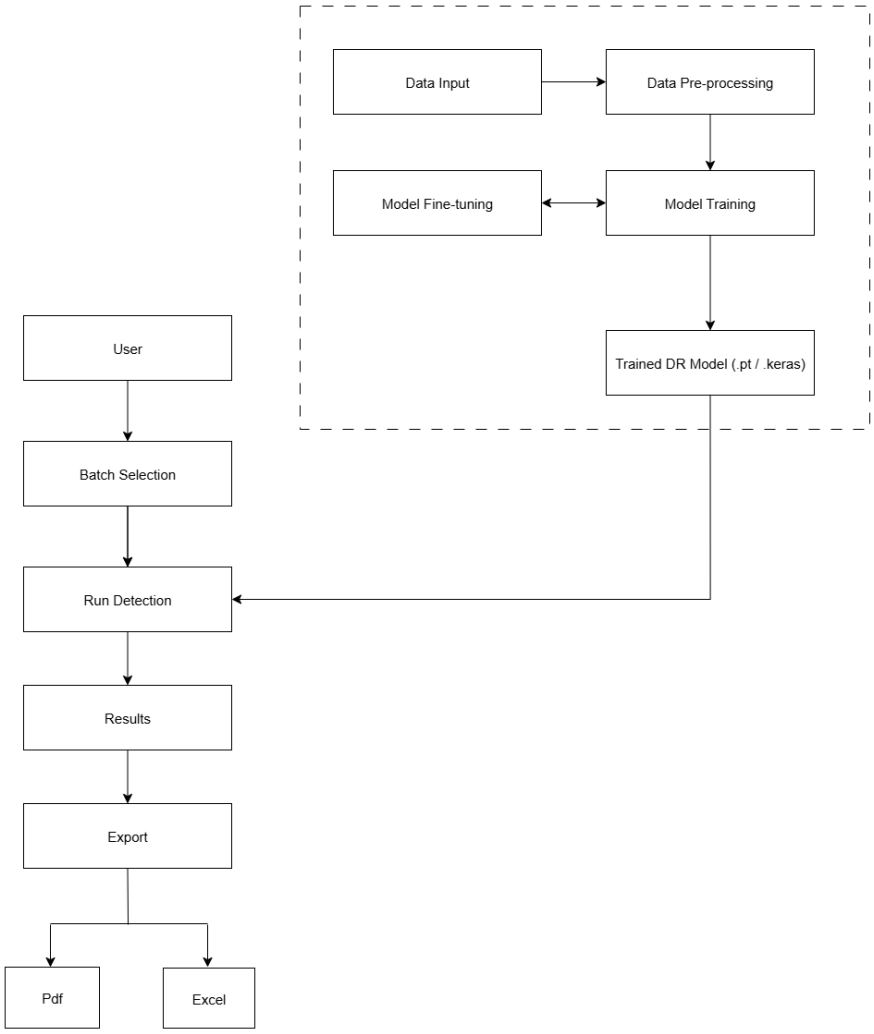


Figure 4.1.1.1 System Architecture Diagram

The Diabetic Retinopathy (DR) detection system integrates the data pipeline, user workflow, detection engine, and reporting module into a desktop application. Each of the component that is involved play an important role to make sure that a smooth execution of training, inference, and result presentation.

First, the data pipeline manages the preparation and training of the DR detection model. It has the subcomponents of “Data Input”, “Data Pre-Processing”, “Model Training”, and “Trained DR Model Export”. The “Data Input” part imports the retinal fundus images from the Kaggle Gaussian-filtered dataset. “Data Pre-Processing” will then resizes those images from the dataset into 224 x 224 pixels, normalizes the pixel values and then applies augmentation to balance minority classes and improve the robustness. “Model Training” part meanwhile will work with the “Model Fine-tune” component so that the model is well trained and performs well when it comes the classification class of DR. “Trained DR Model Export” will then save the .keras or .pt format for the use in the application.

Second, the user workflow part will define how the users interact with the system. It will begin with the “User” who launches the application and enters a UI. Then the user will proceed to “Batch Selection” part to choose retinal image(s) for analysis. This workflow ensures the flexibility when it comes to the clinical scenario where the batch screening is often required.

Third, the detection module will execute the inference process. In the “Run Detection” stage, the selected retinal image(s) will then undergo brief preprocessing (fundus cropping, CLAHE contrast enhancement, resizing to 224×224, and normalization) before passing into the trained DR model. The system will then generate the classification results by classifying each of the image(s) into one of the five DR stages. The outcomes will then be displayed in the next stage which is the “Results” stages that allow the user to get the classification result.

Lastly, the reporting module will be responsible for showing the classification results in app, managing the exports and documentation of the results. It will have some subcomponents like: “PDF Export” and “Excel Export”. The user can see the

results of the classification in the app, but the user also can choose to export the results into PDF or Excel to get a more detailed and clear result. “PDF Export” will generate formatted reports with images, labels, and per-image accuracy, which is suitable for screening records. “Excel Export” creates spreadsheets containing filenames, prediction, and supporting charts for the statistical analysis afterwards.

4.1.2 Use Case Diagram and Description

Figure 4.1.2.1 Case Diagram

The use case diagram above shows how can the user interact with the Diabetic Retinopathy (DR) Detection Desktop Application. The main actor here is User, the person who wants to apply the function of the app to complete his/her task.

The process begins with selecting the retinal images(s), where the user will upload one or more fundus images into the system. Then, the user can try to run the detection. The system will preprocess the image(s) and invoke the trained DR model to map each retinal image into the five classes of DR stages: No DR, Mild, Moderate, Severe and Proliferative DR.

The outcomes will then be displayed on the screen, where the images name, classes and accuracy will be displayed for each of the uploaded images. Then, the user can choose to export the results to PDF or to Excel, providing formatted outputs containing images, predicted labels, accuracy, and statistical charts.

Last but not least, the user can cancel the detection whenever they start the detection in case some incident happens.

4.1.3 Activity Diagram

Figure 4.1.3.1 illustrates the activity diagram of the system. It includes the workflow of the corresponding activities between the “User”, “Diabetic Retinopathy Detection System”, and “Trained Model”.

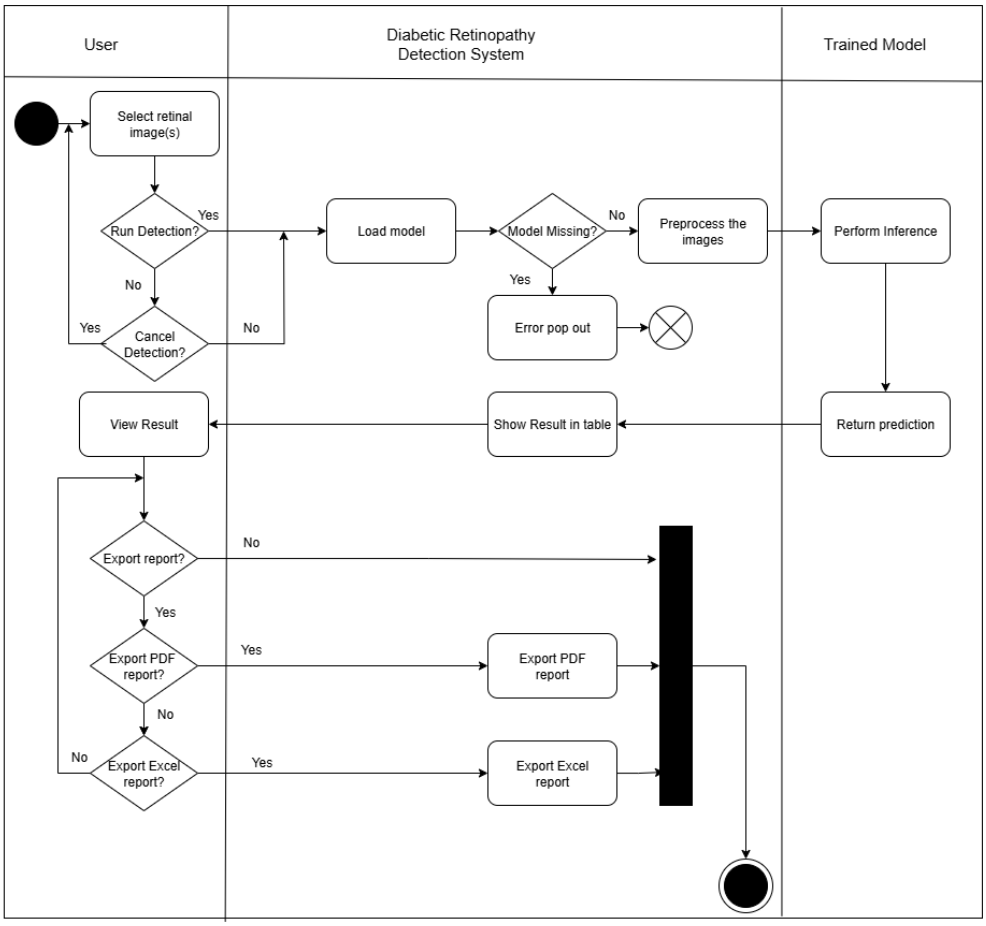


Figure 4.1.3.1 Activity Diagram

4.2 Detailed Explanation on the Development of the System

The development of the diabetic retinopathy desktop application followed a modular approach to ensure reliability, maintainability, and ease of use for end-users.

The system begins with a graphical user interface (GUI) designed to provide a simple and intuitive environment. The users can upload one or multiple retinal image(s) through this interface, and then the images will be queued up to wait for the analysis. To maintain responsiveness, the system incorporates background execution threads that handle intensive tasks such as preprocessing and inference without freezing interface.

Once uploaded, the images will undergo an automated preprocessing stage. At this stage, it will standardize the input by resizing images to 224 x 224 pixels, cropping to focus on the fundus region, enhancing contract by using the CLAHE, and

normalizing the pixel values. These steps replicate the preparation process used during model training, ensuring consistency between the training dataset and the real-world inputs.

After preprocessing, the images are then passed to the inference module, where the DenseNet-201 model executes classification. The model outputs a probability distribution across five stages of diabetic retinopathy (DR), and the class with the highest probability is chosen as the final prediction. This prediction is mapped to a descriptive label and paired with the model's confidence level.

The results are then presented in the results display stage, where users can view predictions directly on the interface. To support documentation and further analysis, the application includes a reporting module that enables users to export results in professional formats. The PDF report includes thumbnails of the images alongside with the predicted labels and confidence values, while the Excel report extends this by including statistical summaries and graphical chats of label distribution.

Finally, the system incorporates configuration management that allows parameters such as labels, reporting titles, and accuracy metrics to be adjusted without modifying the underlying system design. This ensures flexibility for future improvements and adaptations.

Through this development process, the system integrates preprocessing, inference, and reporting into a cohesive workflow. The modular structure ensures that each components operates independently yet contributes to the overall goal of providing a reliable, user-friendly diabetic retinopathy detection tool.

CHAPTER 5

System Implementation

This chapter will show the setting and the configuration of the system, demonstrate on how the system work, the implementation issues and challenges faced during the development and a conclusion remark.

5.1 Settings and Configuration

The diabetic retinopathy desktop application was developed in a reproducible environment with well-defined settings and configuration parameters. This ensures that the system can be consistently installed, executed, and extended by other researchers or developers.

Development Environment

- **Programming Language:** Python 3.10
- **Integrated Development Environment (IDE):** Visual Studio Code (VS Code)
- **Operating System:** Windows 11 (64-bit)
- **Virtual Environment:** Python venv is used to isolate dependencies and manage package versions.

Software Dependencies

All required dependencies are specified in the requirement.txt file to ensure the reproducibility:

- Pysides6 6.7.2
- TensorFlow 2.16.1 (TensorFlow-Intel on Windows)
- Keras 3.3.2
- NumPy 1.26.4

- OpenCV-Python 4.12.0.88
- Pillow 11.3.0
- Scikit-image 0.25.2
- Pandas 2.3.2
- ReportLab 4.4.3
- PyYAML 6.0.2
- Openpyxl 3.1.5

Model Configuration

- **Model Architecture:** DenseNet-201, fine-tuned for diabetic retinopathy detection.
- **Model Export:** Store in keras format for inference compatibility.
- **Input Size:** 224 x 224 pixels, 3-channel RGB
- **Classes:** No_DR, Mild, Moderate, Severe, Proliferative_DR.
- **Per-Class Metrics:** Accuracy values stored in Json to provide contextual information during the result display.

Application Configuration

- **Configuration File:** A YAML configuration file is used to manage the system settings. This file specifies the paths for the trained model, labels, and metrics, as well as the report title and preprocessing options.
- **Labels:** Defined in Json, mapping numerical outputs from the model to descriptive class names.
- **Reports:** Output setting allow for both PDF and Excel formats, with logos automatically detected from the assets folder to ensure branding consistency.
- **Preprocessing Parameter:** Configurable options include enabling/ disabling CLAHE, fundus cropping, and resizing target size (default: 224 x 224).

Execution Workflow

The project is set up to run in three steps:

1. Create and active a Python virtual environment.
2. Install dependencies from requirement.txt
3. Launch the application by executing:

python app/main.py

Output Settings

- **PDF Report:** Includes thumbnails, predicted labels, and accuracy values formatted in a structured layout.
- **Excel Report:** Includes tabular results, summary statistics, and visual charts of class distribution.
- **Storage:** Exported reports are saved to a user-selected location. If a directory is not specified, the system defaults to saving in the current working directory

5.2 System Operation (with Screenshot)

Figure 5.2.1 shows the main interface displayed when the user launches the application (default in light mode)

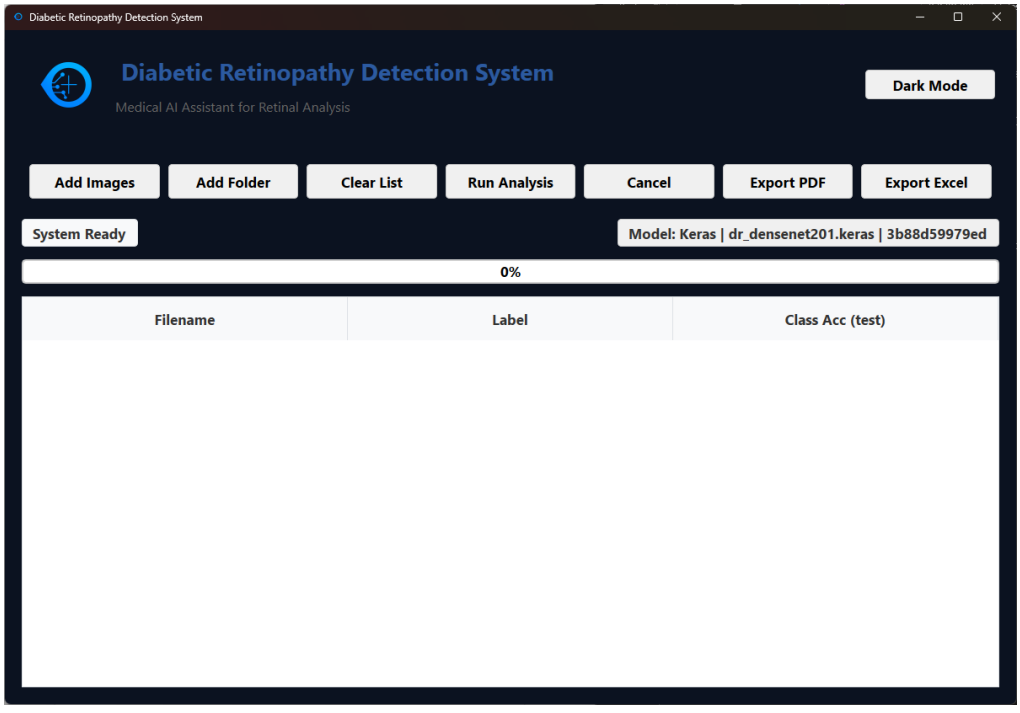


Figure 5.2.1 Main interface of the application

Figure 5.2.2 shows the main displayed when the user uses the light mode/dark mode button

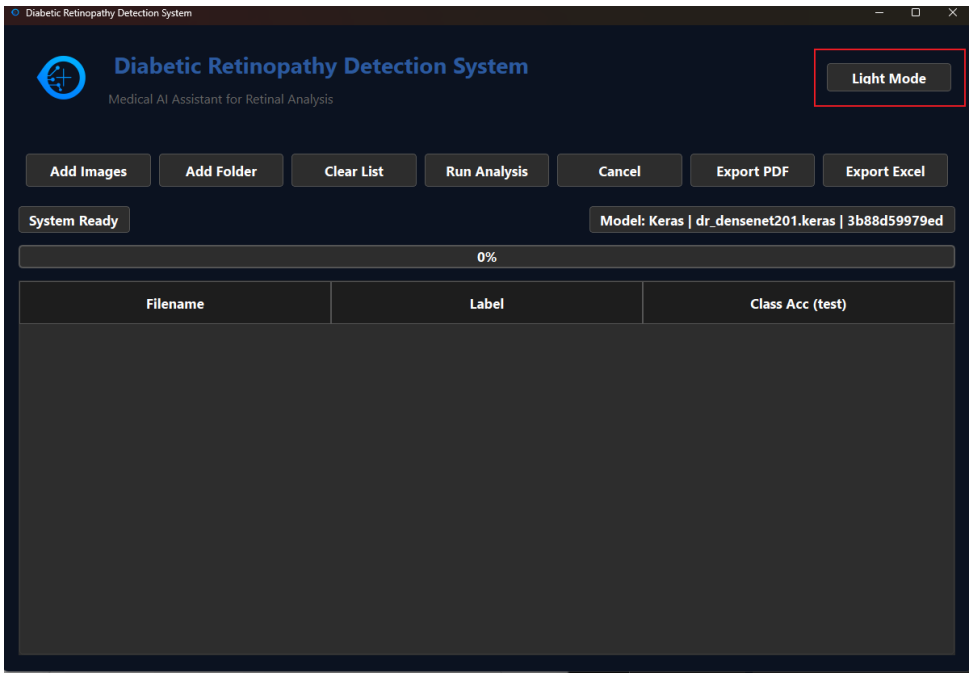


Figure 5.2.2 interface after the user click the light/dark mode button

Figure 5.2.3 to Figure 5.2.5 shows what will happened if the user chooses to upload one retinal image by using the “Add Images” button.

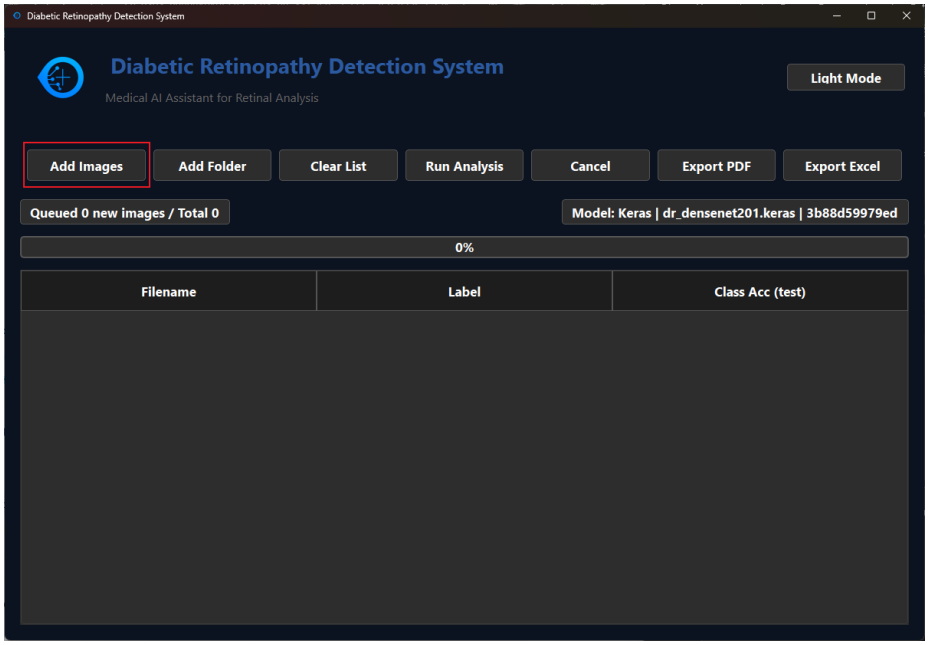


Figure 5.2.3 User click the "Add images" button

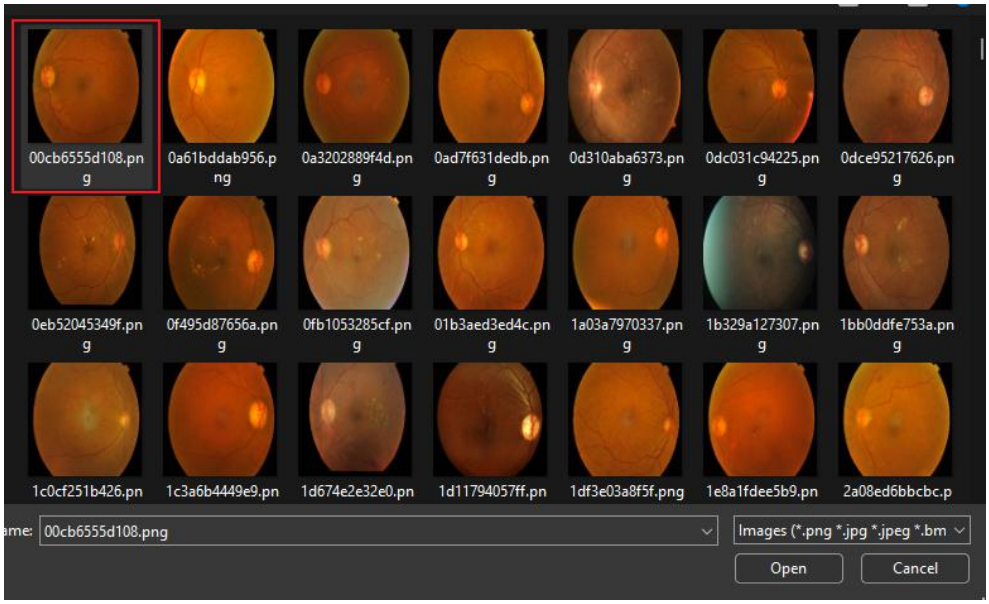


Figure 5.2.4 User choose the retinal images that he/she wants to upload

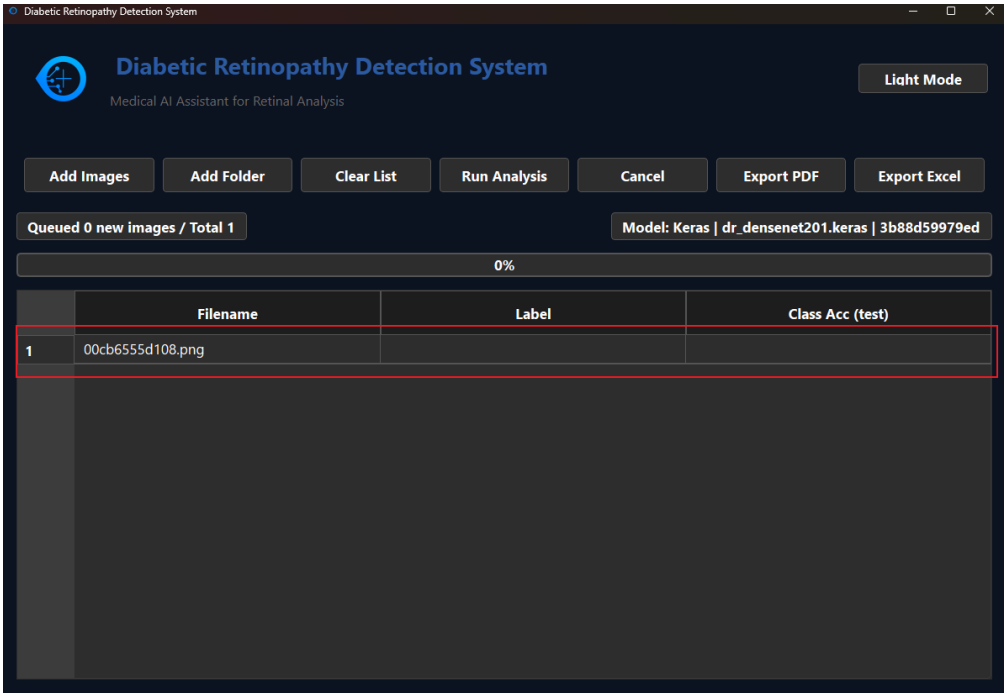


Figure 5.2.5 The images selected is successfully uploaded and ready to go for a detection

Figure 5.2.6 to Figure 5.2.8 will demonstrate what will happened if the user clicks the "Add Folder" button

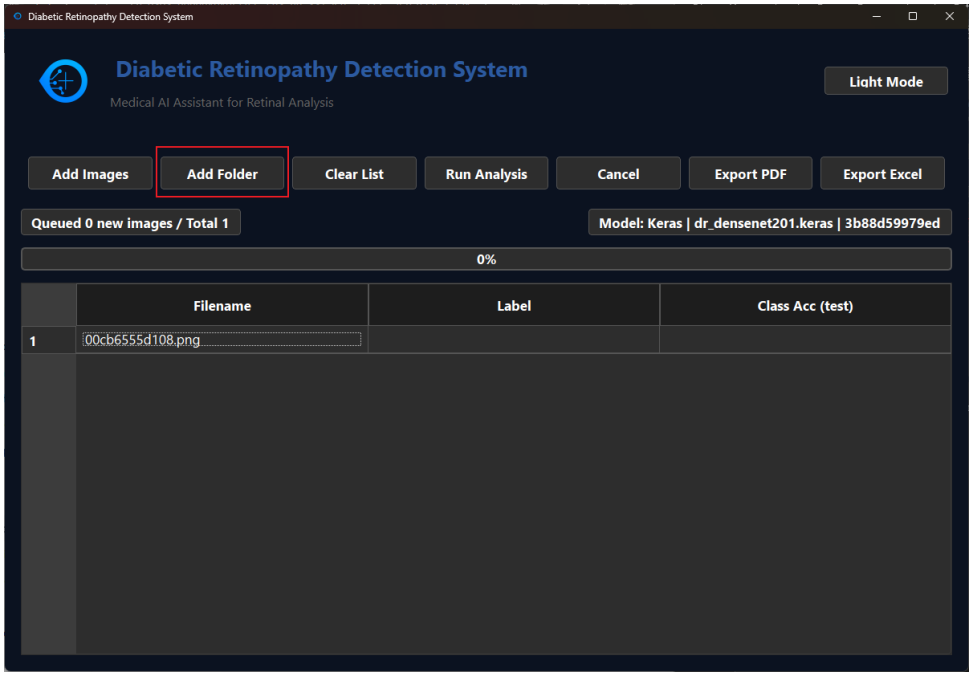


Figure 5.2.6 User click the “Add Folder” button

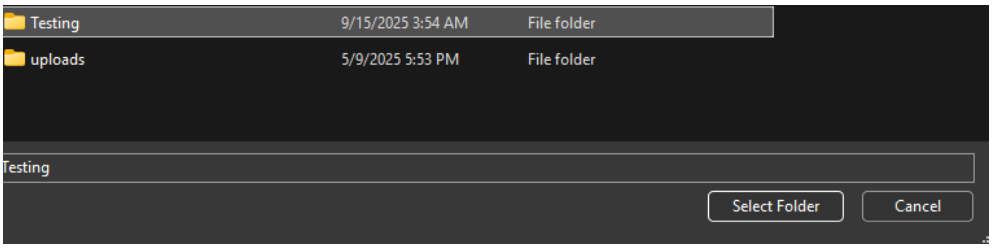


Figure 5.2.7 User choose the folder that he/she wants to upload

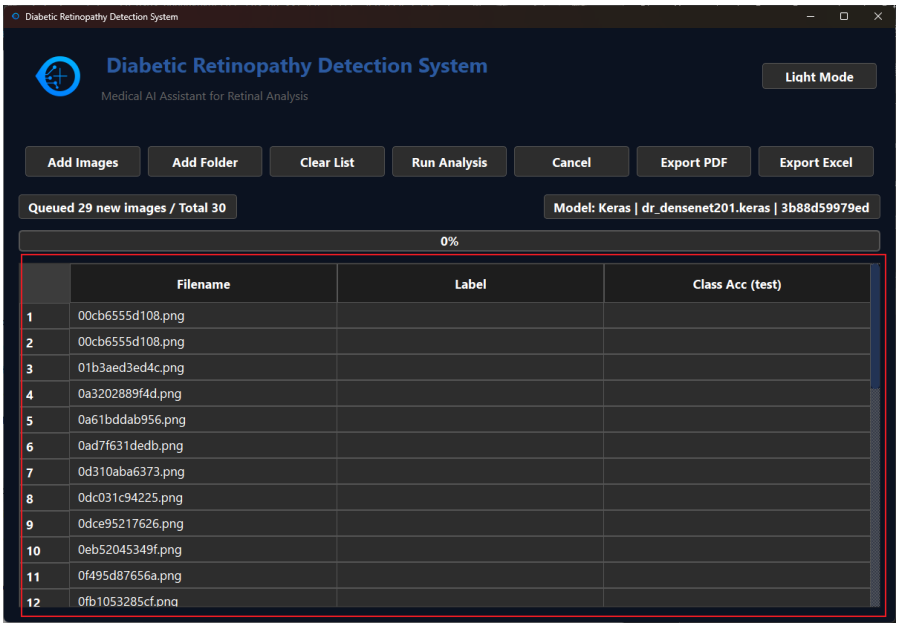


Figure 5.2.8 The images inside the folder will be automatically queued

Figure 5.2.9 to Figure 5.2.10 demonstrate what will happened if the user clicked the “Clear List” button.

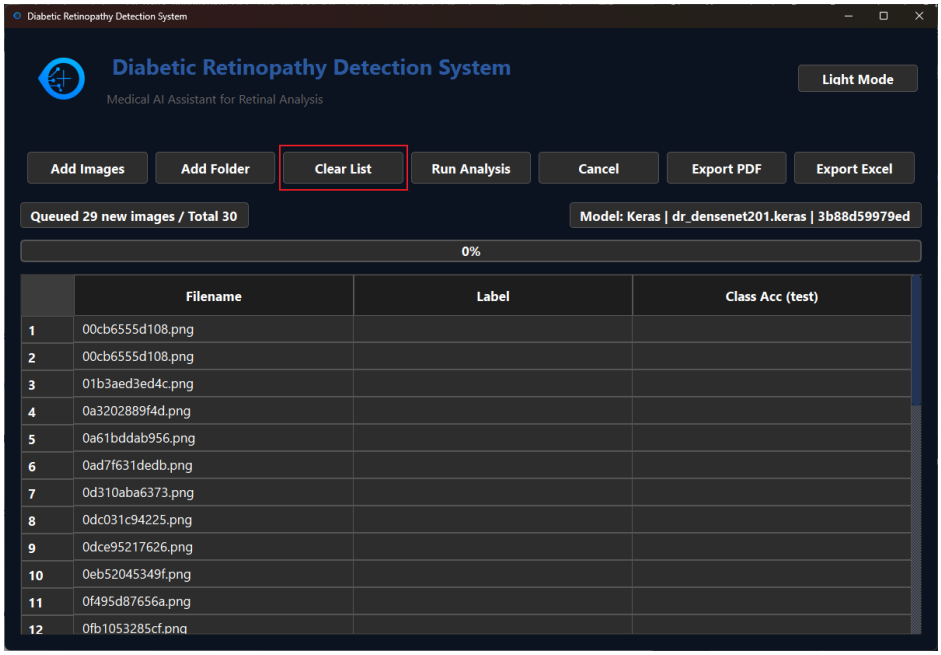


Figure 5.2.9 User clicked the Clear List button

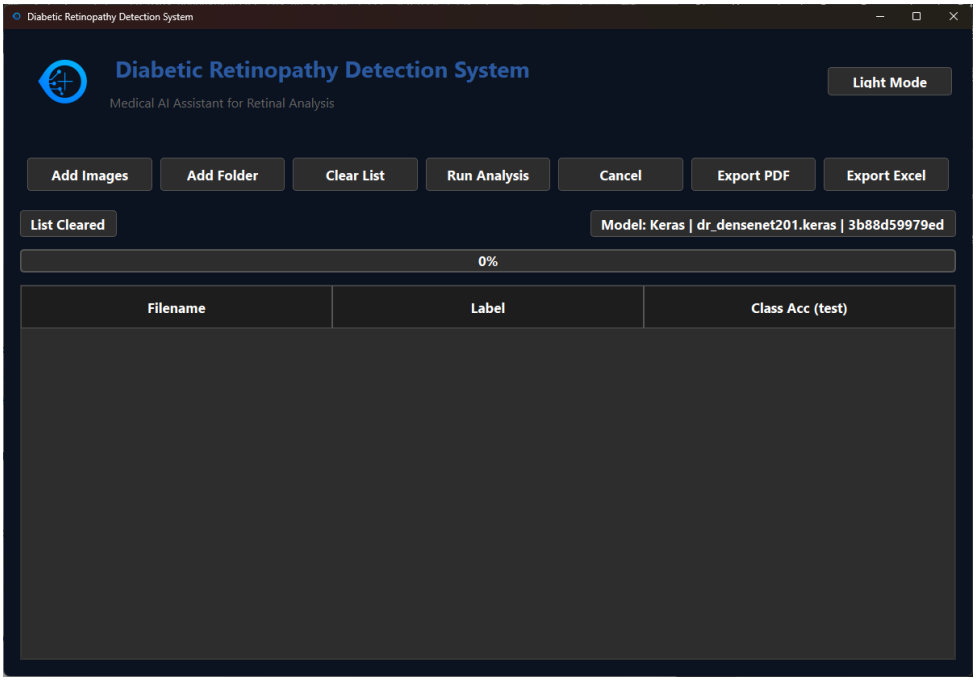


Figure 5.2.10 Uploaded images are all cleared

Figure 5.2.11 to Figure 5.2.12 demonstrates what will happen when user clicked the “Run Analysis" button to analyse the uploaded images

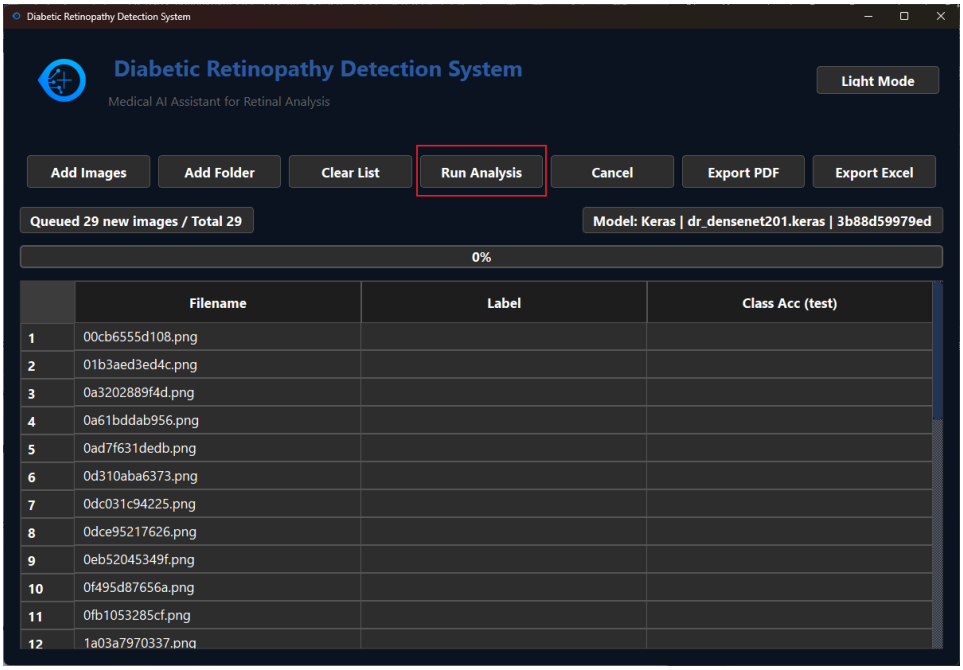


Figure 5.2.11 Users clicked the "Run Analysis" button when there are images uploaded

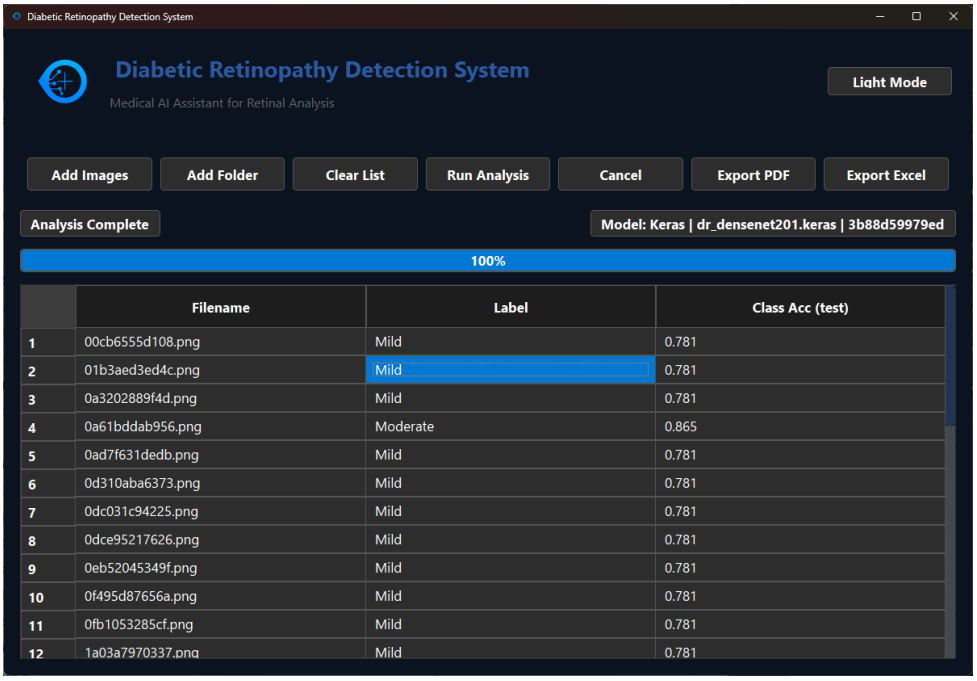


Figure 5.2.12 shows the users the results of the analysis

Figure 5.2.13 to Figure 5.2.14 demonstrates what will happened if the user clicked "Cancel" button whenever the analysis is running.

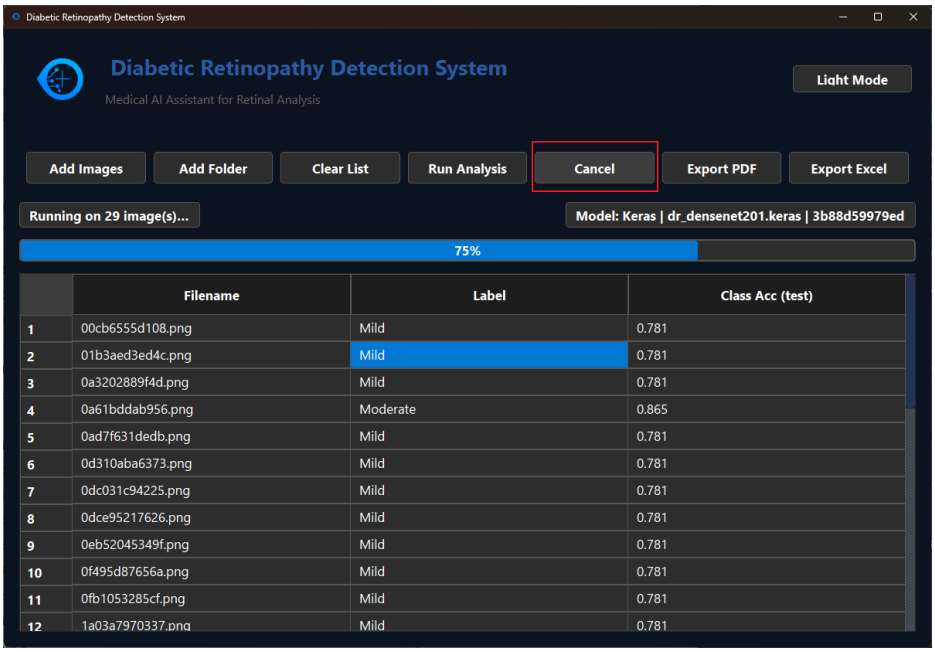


Figure 5.2.13 User clicked the "Cancel" button whenever the analysis in ongoing

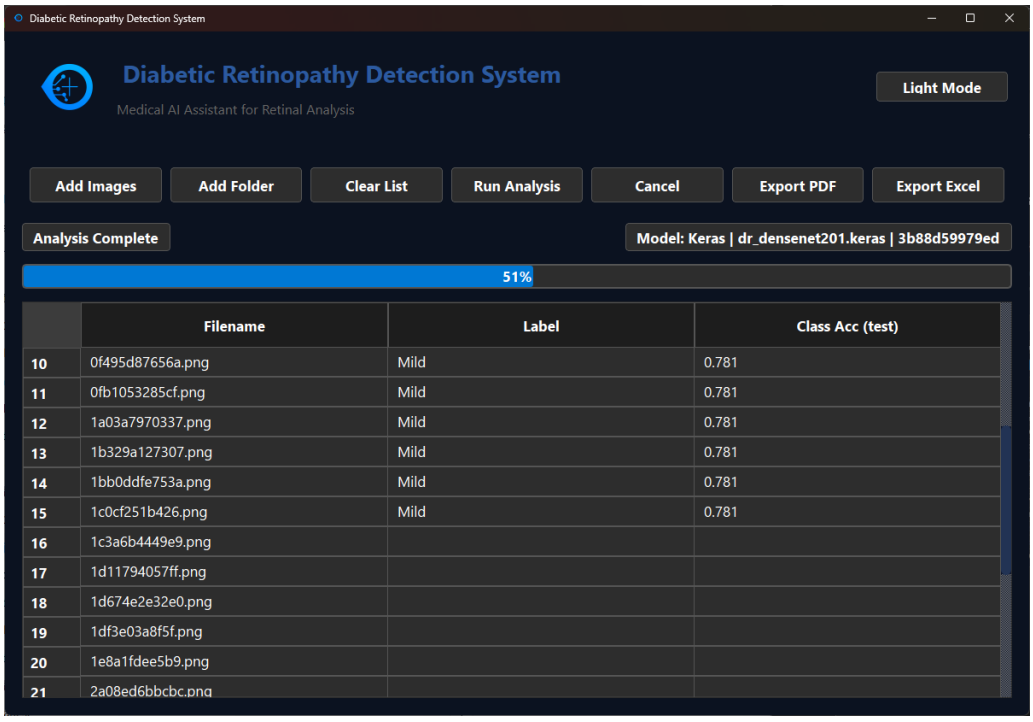


Figure 5.2.14 The analysis is stopped

Figure 5.2.15 to Figure 5.2.17 demonstrated what will happened if the user wishes to export the result into PDF

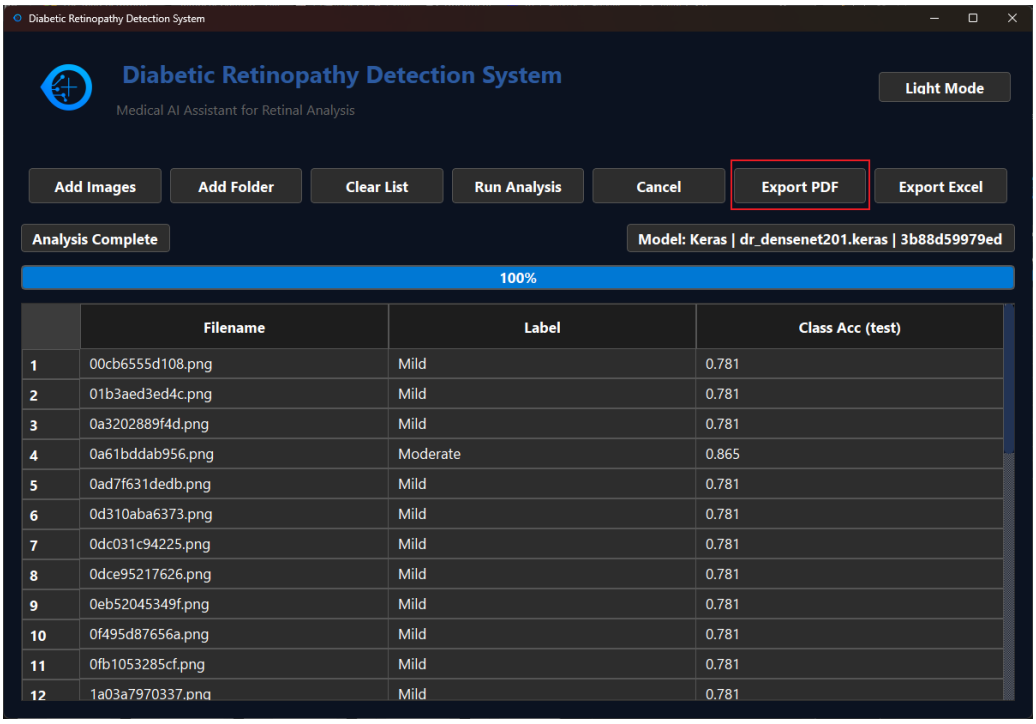


Figure 5.2.15 User clicked the "Export PDF" after the analysis is completed

Figure 5.2.16 User choose where they want to download the pdf file

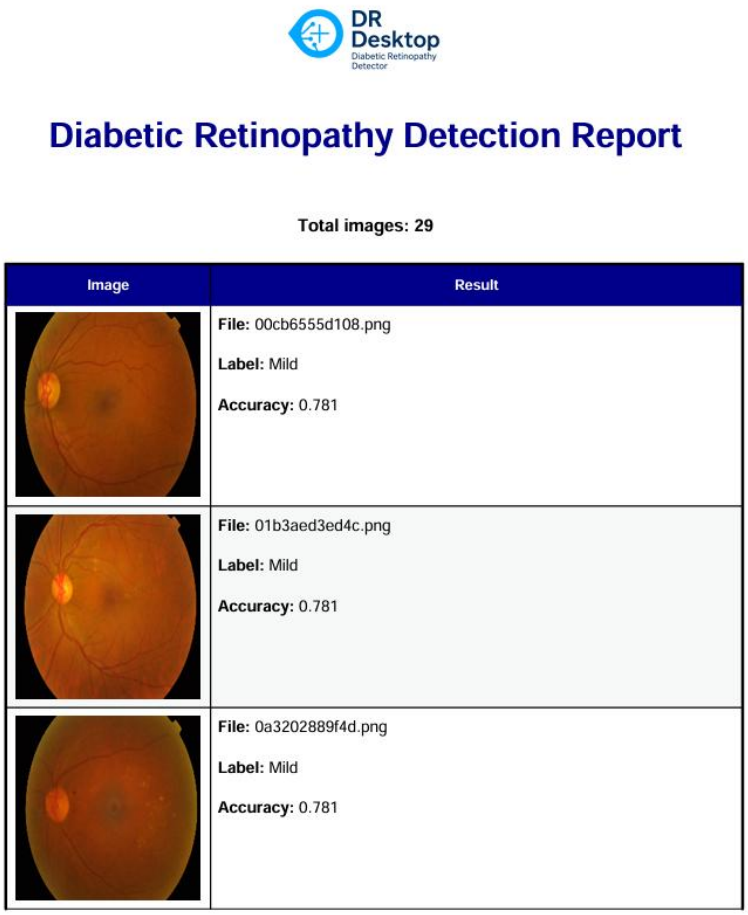
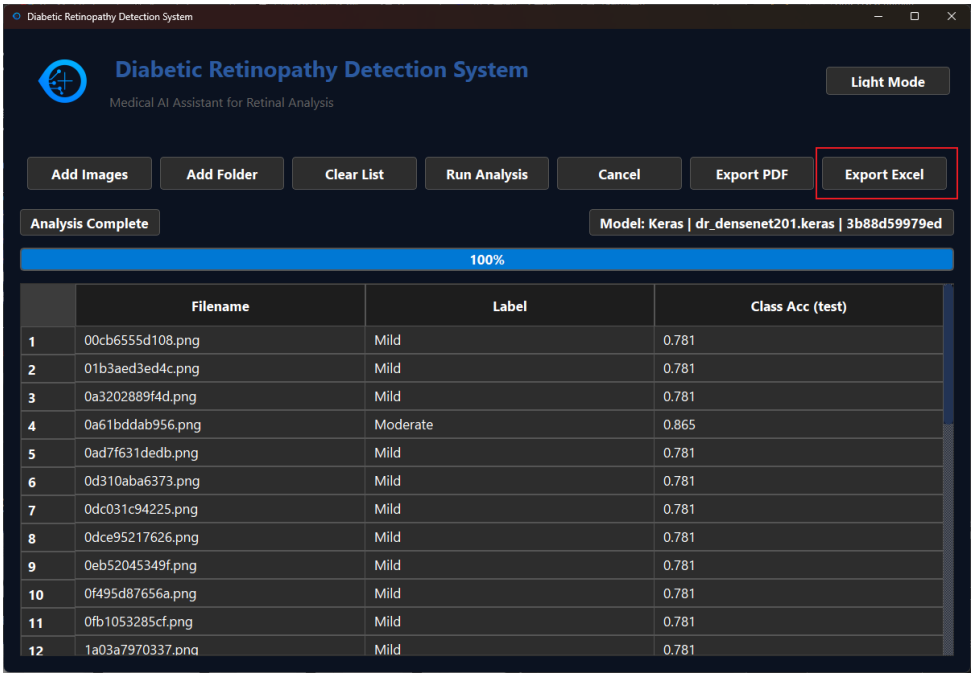


Figure 5.2.17 shows the sample of the PDF

Figure 5.2.18 to Figure 5.2.21 demonstrates the scenario when the user wants to export the result into Excel file.



	Filename	Label	Class Acc (test)
1	00cb6555d108.png	Mild	0.781
2	01b3aed3ed4c.png	Mild	0.781
3	0a3202889f4d.png	Mild	0.781
4	0a61bddab956.png	Moderate	0.865
5	0ad7f631dedb.png	Mild	0.781
6	0d310aba6373.png	Mild	0.781
7	0dc031c94225.png	Mild	0.781
8	0dce95217626.png	Mild	0.781
9	0eb52045349f.png	Mild	0.781
10	0f495d87656a.png	Mild	0.781
11	0fb1053285cf.png	Mild	0.781
12	1a03a7970337.png	Mild	0.781

Figure 5.2.18 User clicked the "Export Excel" after the analysis is completed

Figure 5.2.19 User choose where they want to download the excel file

	A	B	C	D
1	Filename	Label	Accuracy	Image Path
2	00cb6555d108.png	Mild	0.781	C:/FYP/Testing\00cb6555d108.png
3	01b3aed3ed4c.png	Mild	0.781	C:/FYP/Testing\01b3aed3ed4c.png
4	0a3202889f4d.png	Mild	0.781	C:/FYP/Testing\0a3202889f4d.png
5	0a61bddab956.png	Moderate	0.865	C:/FYP/Testing\0a61bddab956.png
6	0ad7f631dedb.png	Mild	0.781	C:/FYP/Testing\0ad7f631dedb.png
7	0d310aba6373.png	Mild	0.781	C:/FYP/Testing\0d310aba6373.png
8	0dc031c94225.png	Mild	0.781	C:/FYP/Testing\0dc031c94225.png
9	0dce95217626.png	Mild	0.781	C:/FYP/Testing\0dce95217626.png
10	0eb52045349f.png	Mild	0.781	C:/FYP/Testing\0eb52045349f.png
11	0f495d87656a.png	Mild	0.781	C:/FYP/Testing\0f495d87656a.png
12	0fb1053285cf.png	Mild	0.781	C:/FYP/Testing\0fb1053285cf.png
13	1a03a7970337.png	Mild	0.781	C:/FYP/Testing\1a03a7970337.png
14	1b329a127307.png	Mild	0.781	C:/FYP/Testing\1b329a127307.png
15	1bb0ddfe753a.png	Mild	0.781	C:/FYP/Testing\1bb0ddfe753a.png
16	1c0cf251b426.png	Mild	0.781	C:/FYP/Testing\1c0cf251b426.png
17	1c3a6b4449e9.png	Mild	0.781	C:/FYP/Testing\1c3a6b4449e9.png
18	1d11794057ff.png	Mild	0.781	C:/FYP/Testing\1d11794057ff.png
19	1d674e2e32e0.png	Mild	0.781	C:/FYP/Testing\1d674e2e32e0.png
20	1df3e03a8f5f.png	Mild	0.781	C:/FYP/Testing\1df3e03a8f5f.png
21	1e8a1fdee5b9.png	Mild	0.781	C:/FYP/Testing\1e8a1fdee5b9.png
22	2a08ed6bbcbc.png	Moderate	0.865	C:/FYP/Testing\2a08ed6bbcbc.png
23	2a8a9e957a6c.png	Mild	0.781	C:/FYP/Testing\2a8a9e957a6c.png
24	2d7666b8884f.png	Mild	0.781	C:/FYP/Testing\2d7666b8884f.png

DR Detection Results | Charts & Statistics

Figure 5.2.20 The first sheet of the excel file

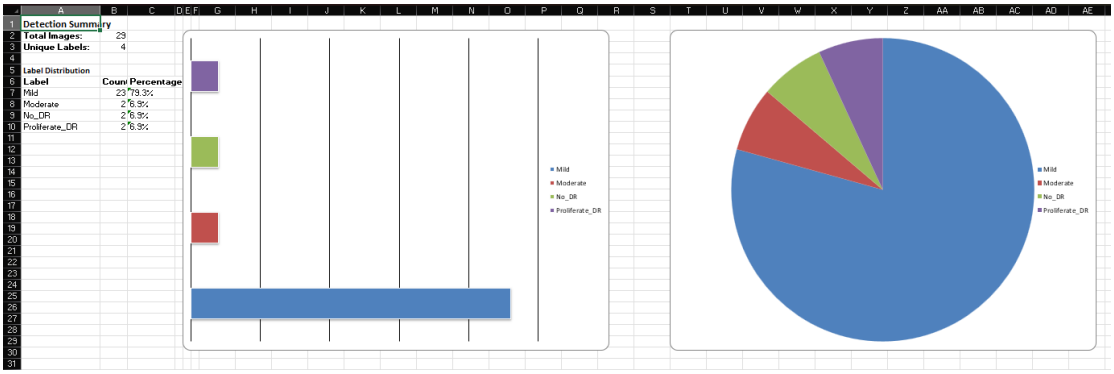


Figure 5.2.21 The second sheet of the excel file

5.3 Implementation Issues and Challenges

During the development and implementation of the diabetic retinopathy desktop application, several issues and challenges were encountered. These challenges arose from the environment dependencies, model integration, system performance, and user interface design. It is a must to ensure that these challenges to be overcome so that the application functions as a reliable and user-friendly tool.

One of the challenges was ensuring compatibility between different software libraries. TensorFlow, Keras, and NumPy versions frequently conflicted during detection, resulting in happening the errors when attempting to load the model file in keras format. To resolve this, the system was standardized on TensorFlow 2.16.1 with Keras 3.3.x, while the NumPy was fixed below version 2.0 to avoid incompatibility. A dedicated virtual environment was also used to isolate the dependencies and avoid package version drift.

Running detection directly on the main application thread initially caused the interface to freeze during inference. This issue was resolved by implementing background worker threads for model execution, keeping the interface responsive and allowing the users to cancel the tasks.

5.4 Conclusion Remark

In conclusion, the system implementation of the diabetic retinopathy (DR) desktop application has successfully transformed the proposed methodology and design into a

working solution. Through careful configuration of the development environment, integration of the trained DenseNet-201 model, and the incorporation of preprocessing and reporting modules, the application can provide accurate and user-friendly diabetic retinopathy detection. The graphical user interface enables intuitive interface for end-users, while the reporting function ensures results can be professionally documented and shared.

Although several implementation issues and challenges were encountered, such as dependency management and interface responsiveness. These challenges have then been overcome systematically to improve the stability and usability of the system. The final system shows that not only technical feasibility but also practical functionality, making it a reliable tool for automated DR screening. This successful implementation lays the groundwork for future enhancements in subsequent chapter.

CHAPTER 6

System Evaluation and Discussion

This chapter will show the system testing and performance metrics, testing setup and results, project challenges, objectives evaluation and a conclusion remark.

6.1 System Testing and Performance Metrics

System testing was designed to isolate the effect of evaluation methodology while keeping the modelling choices identical to the referenced work. First, we reproduced the same network configuration, loss, input resolution, and preprocessing as the article so that the architectural factors would not confound the comparison. The only methodological departure in this replica was to adopt the same stratified train-validation-test split protocol that is used in this paper which are 70%,20%,10% for each of the set, yielding a fixed held-out test partition. This single change ensures that the test set remains constant across the runs and avoid any chance of mixing training images into evaluation, and enables a fair, apple-to-apple comparison against our implementation.

The system testing for this paper also proceed in two stages to demonstrate the benefit of the model selection by validation. This paper evaluated both last epoch model and the best-checkpoint model. This is due to both evaluations use the same fixed test set, any difference in results reflects genuine generalization differences rather than changes in sample composition. Reporting both outcomes makes clear whether relying on the final epoch is optimal, or whether checkpointing shows a stronger model for deployment for the system.

Performance was assed using clinically relevant, class-aware metrics. We report accuracy as the overall proportional of the correct prediction; precision, recall and F1-score for each diabetic retinopathy grade; and their macro and weighted averages to summarize performance under class imbalance. In addition, we computer the per-class accuracy at the decision level. Practically, this is read from the row-normalized confusion matrix, where each row sums to one and the diagonal entries corresponding

to per-class recall (sensitivity) which is normally used as per-class accuracy in multi-class screening contexts. For completeness, we present two confusion matrices. One of it in the raw matrix in counts which shows the absolute error distribution across the classes, meanwhile the other one is the normalized matrix, which highlights per-class correctness rates independent of class size. These metrics are produced on the fixed test set for both the last-epoch and best-checkpoint models and are retained for export within the application's reporting module.

6.2 Testing Setup and Result

This study uses the Diabetic Retinopathy 224 x 224 Gaussian-Filtered dataset from Kaggle. All the images were resized to 224 x 224 normalized to [0,1] and labelled into five classes (No_DR, Mild, Moderate, Severe, Proliferative DR). A stratified 70%/20%/10% train-validation-test split with a fixed seed was adopted, showing a single, isolated test set.

To establish a fair baseline against the reference work [16], we first adopt and produced an article-replica run in which the model, loss, input resolution and preprocessing were kept exactly the same as in the article. The only deviation was methodological. Instead of drawing a random subset of 500 images, we evaluated on the fixed 637-image test set from the split above. Under this condition the article replica have achieved an accuracy of 0.73, precision of 0.77 and a F1-score of 0.72. The raw and normalized confusion matrices showed poor recognition of Severe and Proliferative DR, with low row-diagonal values for these classes, confirming that the higher accuracy reported by the article on a random 500- image sample does not persist when tested on a true held-out set.

```
Accuracy Score : 0.7319223985890653
Precision Score : 0.7655854403949852
F1 Score : 0.7157160329658863
```

Figure 6.2.1 Result of article method

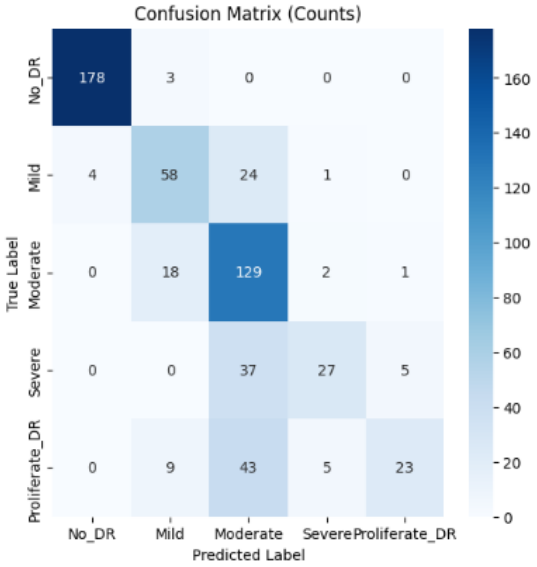


Figure 6.2.2 Confusion matrix for article method

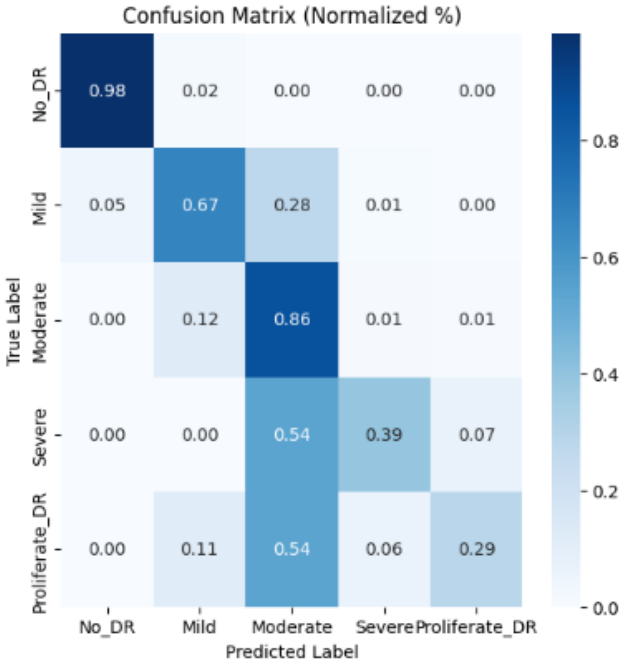


Figure 6.2.3 Normalized Confusion matrix for article method

Then the results of the model that is implemented in this paper is reported as well. This paper applied the improvement used in our system. The last-epoch model attained a 0.77 – 0.78 accuracy, precision of 0.82 and F1-score of 0.78. Its normalized confusion matrix highlighted a better prediction on Severe and Proliferative DR classes.

```
Accuracy Score : 0.7786499215070644
Precision Score : 0.8220917860795495
F1 Score : 0.7841685923693426
```

Figure 6.2.4 Results of last epoch model

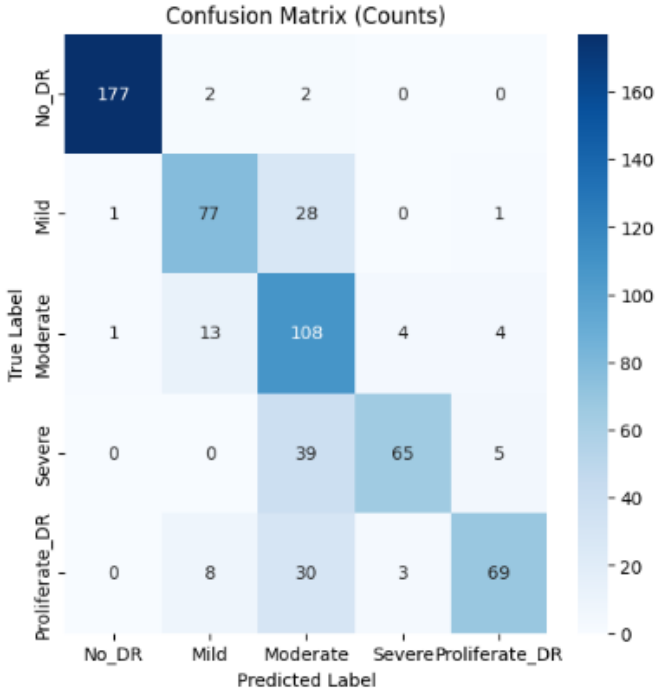


Figure 6.2.5 Confusion matrix of last epoch model

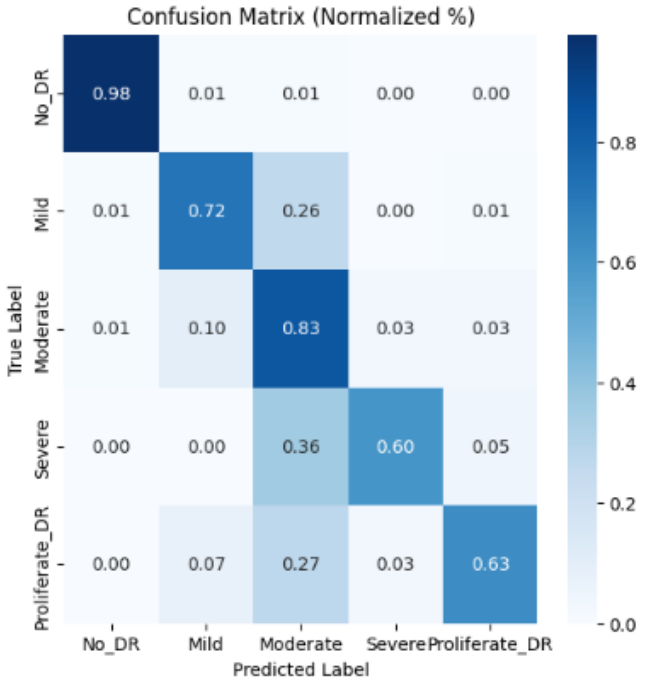


Figure 6.2.6 Normalize matrix of last epoch model

The results become even better when it comes to evaluate the best-checkpoint model which is saved at the peak validation accuracy. It further increased the performance to 0.81 accuracy, 0.81 precision, F1-score of 0.81. Its normalized confusion matrix showed a recall of 0.72 for Severe classes which is higher than the last epoch model, indicating a better generalization across the classes.

```
=== Evaluation using best checkpoint weights ===
Accuracy Score : 0.8100470957613815
Precision Score : 0.8120826444994771
F1 Score : 0.808385075452674
```

Figure 6.2.7 Results of best checkpoint model

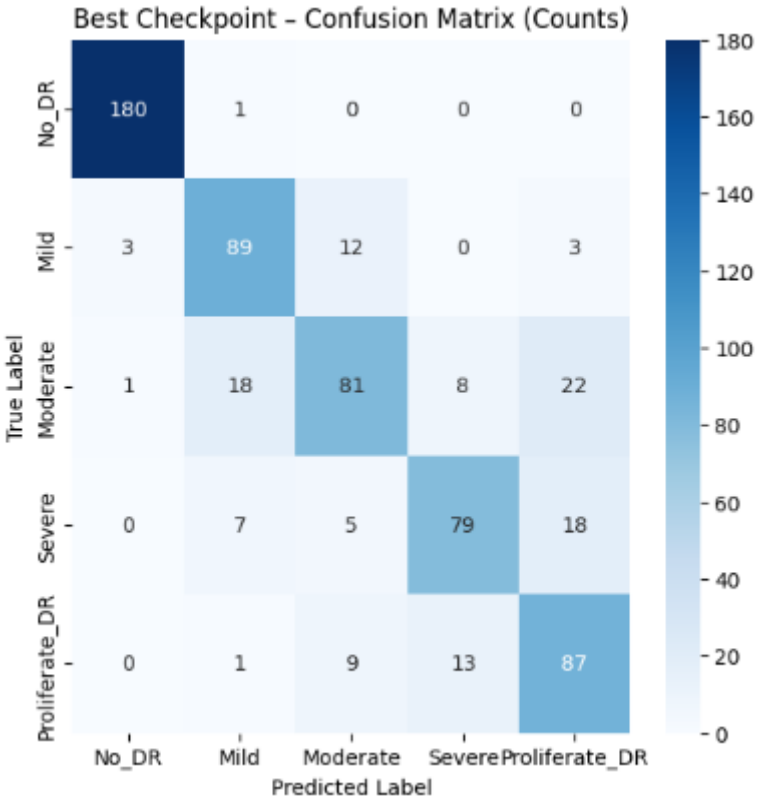


Figure 6.2.8 Confusion matrix of best checkpoint model

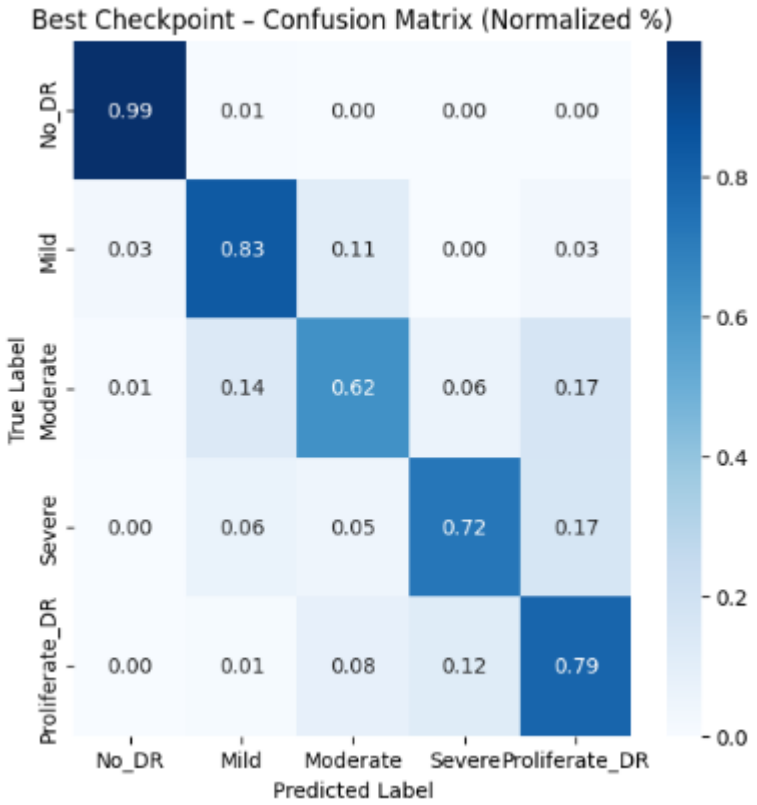


Figure 6.2.9 Normalized confusion matrix of best checkpoint model

Finally, after evaluating the article-replica (0.73 accuracy), last epoch model (0.78 accuracy), and our best checkpoint model (0.81 accuracy) that were carried out on the isolated testing set. These results show that our approach outperforms the article's method even before the best checkpointing, and after adopting the best checkpoint, it shows the strongest and most reliable configuration for implementation.

6.3 Project Challenges

There were a few practical challenges were faced during the project development. First is for the reference article evaluate their performance using a random 500-image subset, which could accidentally include the training images and inflate results. We then replaced this with stratified 70/20/10 train -validate-test split.

Second, the dataset is suffered from a serious class imbalance problem, which fit the actual clinical scenario where Severe, Mild and Proliferate DR is under-represented. Class weight that is adopted in paper [6] also been adopted in this paper to overcome the class imbalance issue in the dataset.

Finally, the article exported weights-only files, which did not integrate cleanly with the system that is developed in this paper, because the app needs a single, self-contained model file. To make the deployment reliable, we switched to Keras model format, which bundles the architecture and weights in one file and loads directly in the application.

6.4 Objective Evaluation

i. To develop an automated DR screening system using deep learning techniques.

This project successfully developed an automated screening system based on DenseNet-201 backbone with transfer learning. The system ingests retinal fundus images, performs standardized preprocessing (224 x 224, normalization), and classifies them into five DR stages without manual feature engineering. Two model snapshots were assessed on the same test set in which the last epoch model has a 0.77 – 0.78 accuracy, and the beast check-point model have a 0.81 accuracy. The checkpointed model showed a better generalization and more balanced per-class results. Therefore, the objective of building an automated deep-learning DR screener was achieved.

ii. To overcome dataset class imbalance through augmentation and class weighting.

The dataset is skewed toward No DR/ Moderate DR, where other class are underperformed. Targeted data augmentation was applied to strengthen minority-class representation and reduce overfitting. The pipeline also supports class weighting and threshold calibration. Both the last epoch model and the best checkpoint model outperformed the reference work article and show a more balance results across all class show a better generalization across the class show class imbalance is overcome. This objective is achieved.

iii. To implement a desktop-based application for user-friendly DR screening.

This project delivered a Windows desktop-based application that allows users to upload single images or folders, run batch detection, and obtain instant DR grades alongside exportable reports in PDF and Excel formats. The model is

packaged as single Keras file replacing the weight-only artifacts from the article that did not integrate cleanly with the app. The application also surfaces per-class metrics, aligning evaluation output with clinical reporting needs. User-friendly desktop solution was achieved.

6.5 Concluding Remark

This chapter establish a rigorous and reproducible basis for evaluating the proposed DR screening system. By holding the model configuration consistent with the reference work, we change the evaluation protocol to a stratified 70/20/10 split, we removed the risk of data leakage inherent in random sub-sampling and enable an honest, like-for-like comparison. The article-replica achieved an accuracy of 0.73, our last-epoch model improved this into 0.78 and the best-checkpoint model reached 0.81 accuracy, with clearer diagonals in the normalized confusion matrices especially for the clinically important minority classes. These results demonstrate that validation-based checkpointing offers better generalization than relying on the final epoch and that the corrected evaluation protocol yields trustworthy estimates of true performance.

Practical challenges encountered dataset imbalance, reproducible splitting, and deployment format were addressed through targeted augmentation and class weighting. Collectively, overcoming these challenges improve the sensitivity where it matters, make results repeatable, and ensure the model loads reliably in the delivered system. Overall, the objectives of this paper were met.

CHAPTER 7

Conclusion and Recommendations

This chapter will discuss about conclusions and the recommendations for future works.

7.1 Conclusion

This project successfully developed a desktop-based diabetic retinopathy (DR) screening system that automates fundus-images grading with deep learning. The solution couples a fine-tuned DenseNet-201 backbone with a reproducible training-evaluation workflow (stratified 70/20/10 split; fixed held-out test set), and a Windows desktop UI that supports single and batch inference where the results can be exportable into PDF or Excel format reports.

The system also overcomes some common constraint in medical imaging which are limited labels and class imbalance by using transfer learning, targeted data augmentation, and class weighting without changing the core architecture. The desktop application operationalizes the trained model into a usable tool, where the user can choose to upload one image or entire folder, run batch classification, and generate a clinician-friendly summaries.

System evaluation adopted common performance metrics, which is confusion matrix, precision, recall, F1-score on a test set. The findings highlight strong overall performance and also surface the know challenge of minority-class recall for Severe and Proliferative DR, which is consistent with the dataset's imbalance. This paper documents both last-epoch and best-checkpoint performance on an identical test set to transparently demonstrated the benefit of validation-guided model selection.

In conclusion, this paper delivers a complete, reproducible deep-learning pipeline and a functioning desktop application for automated DR-screening. It demonstrates that modern CNN-based methods can be translated into a practical tool with consistent evaluation and traceable results.

7.2 Recommendation

Despite meeting the project objectives, the systems can be improved in several practical areas. Below are concise recommendations that can be applied to this desktop DR screening app.

- **Speed Up Inference**

  Offer a “Fast Mode” that swaps the DenseNet-201 for a lighter backbone such as MobileNet or EfficientNet-S to improve throughput on CPU-only machines. The user can choose to use a the “Fase Mode” or “Accurate Mode” in Settings based on their hardware and throughput needs.

- **Balance the Data**

  Improve class balance, especially when it comes to Severe, Mild and Proliferative DR, by targeted data collection and clinically realistic augmentation to reduce bias toward majority classes.

- **Import Model**

  May enable the user import model that they more preferable to the system and they can then choose to use the model that they prefer to be used for the detection in the settings.

- **Saved & Load settings**

  Enable the user to save and export their settings to a small file or enable the user to import the app settings. This may help if the user wishes to run the app in another device without a complicated and troublesome resetting all the app settings again.

REFERENCES

[1] Nikos Tsiknakis *et al.*, “Deep learning for diabetic retinopathy detection and classification based on fundus images: A review,” *Computers in Biology and Medicine*, vol. 135, pp. 104599–104599, Jun. 2021, doi: https://doi.org/10.1016/j.compbiomed.2021.104599.

[2] S. R. Rath, “Diabetic Retinopathy 224x224 Gaussian Filtered,” *Kaggle.com*, 2019. https://www.kaggle.com/datasets/sovitrath/diabetic-retinopathy-224x224-gaussian-filtered (accessed Sep. 18, 2025).

[3] J. W. Y. Yau *et al.*, “Global Prevalence and Major Risk Factors of Diabetic Retinopathy,” *Diabetes Care*, vol. 35, no. 3, pp. 556–564, Feb. 2012, doi: https://doi.org/10.2337/dc11-1909.

[4] M. D. Abramoff, M. K. Garvin, and M. Sonka, “Retinal Imaging and Image Analysis,” *IEEE Reviews in Biomedical Engineering*, vol. 3, pp. 169–208, 2010, doi: https://doi.org/10.1109/rbme.2010.2084567.

[5] M. D. Abràmoff, P. T. Lavin, M. Birch, N. Shah, and J. C. Folk, “Pivotal trial of an autonomous AI-based diagnostic system for detection of diabetic retinopathy in primary care offices,” *npj Digital Medicine*, vol. 1, no. 1, Aug. 2018, doi: https://doi.org/10.1038/s41746-018-0040-6.

[6] H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding, and Y. Zheng, “Convolutional Neural Networks for Diabetic Retinopathy,” *Procedia Computer Science*, vol. 90, pp. 200–205, 2016, doi: https://doi.org/10.1016/j.procs.2016.07.014.

[7] V. Gulshan *et al.*, “Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs,” *JAMA*, vol. 316, no. 22, p. 2402, Dec. 2016, doi: https://doi.org/10.1001/jama.2016.17216.

[8] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | Saturn Cloud Blog,” *saturncloud.io*, Dec. 15, 2018. https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/ (accessed Sep. 10, 2024).

[9] H.-C. Shin et al., “Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning,” IEEE

Transactions on Medical Imaging, vol. 35, no. 5, pp. 1285–1298, May 2016, doi: https://doi.org/10.1109/tmi.2016.2528162.

[10] I. Kandel and M. Castelli, “Transfer Learning with Convolutional Neural Networks for Diabetic Retinopathy Image Classification. A Review,” Applied Sciences, vol. 10, no. 6, p. 2021, Mar. 2020, doi: https://doi.org/10.3390/app10062021.

[11] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, “Transfusion: Understanding Transfer Learning for Medical Imaging,” Neural Information Processing Systems, 2019. https://proceedings.neurips.cc/paper_files/paper/2019/hash/eb1e78328c46506b46a4ac4a1e378b91-Abstract.html

[12] jindongwang, “transferlearning/doc/迁移学习简介.md at master • jindongwang/transferlearning,” GitHub, 2017. https://github.com/jindongwang/transferlearning/blob/master/doc/%E8%BF%81%E7%A7%BB%E5%AD%A6%E4%B9%A0%E7%AE%80%E4%BB%8B.md (accessed Sep. 10, 2024).

[13] R. Wattamwar, R. Thopate, M. Mohite, M. Daga, and P. Nile, “Detection of Diabetic Retinopathy using CNN,” *International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)*, vol. 11, no. 5, pp. 81–87, May 2024. [Online]. Available: https://ijercse.com/article/20%20May%202024%20IJERCSE.pdf

[14] P. S. Ezekiel, O. E. Taylor, and F. B. Deedam-Okuchaba, “A Deep Learning Approach in Detecting Diabetic Retinopathy Using Convolutional Neural Network on Gaussian Filtered Retina Scanned Images,” International Journal of Computer Sciences and Engineering (IJCSE), vol. 8, no. 6, pp. 34–39, Jun. 2020, doi: 10.26438/ijcse/v8i6.3439.

[15] H. Verma et al., “Detecting Diabetic Retinopathy using Deep Learning,” Digital Digest (SiliconTech), vol. 24, pp. 3–6, Sep.–Nov. 2020. [Online]. Available: https://silicon.ac.in/wp-content/uploads/2021/05/Digital-Digest-Sep-2020-Nov-2020.pdf

[16] B. Jeganathan, “Diabetic Retinopathy Detection Using Deep Learning on Gaussian-Filtered Retina Images,” IRE Journals, vol. 8, no. 4, pp. 171-179, Oct. 2024. [Online]. Available: https://www.irejournals.com/formatedpaper/1706392.pdf

[18] Python Software Foundation, “The Python Language Reference,” Python 3 Documentation, 2025. [Online]. Available: https://docs.python.org/3/reference/ (accessed Sep. 20, 2025).

[19] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, ... and S. J. van der Walt, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” Nature Methods, vol. 17, no. 3, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[20] W. McKinney, “Data Structures for Statistical Computing in Python,” in Proc. 9th Python in Science Conf. (SciPy 2010), 2010, pp. 51–56. doi: 10.25080/Majora-92bf1922-00a.

APPENDICES

A.1 Poster