FINAL PROJECT - ES234849

DEEP LEARNING-DRIVEN DIABETIC RETINOPATHY CLASSIFICATION: FROM OPTIMIZED CNN TO FLASK-BASED DEPLOYMENT

Reyhan Emeraldo

NRP 5026201095

Advisor

Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D.
NIP 1988201812010

Bachelor of Information System

Departemen of Information System

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2025

TUGAS AKHIR – ES234849

KLASIFIKASI RETINOPATI DIABETIK BERBASIS DEEP LEARNING: DARI CNN YANG DIOPTIMALKAN HINGGA DEPLOYMENT BERBASIS FLASK

Reyhan Emeraldo

NRP 5026201095

Dosen Pembimbing

Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D.

NIP 1988201812010

Program Studi Sarjana Sistem Informasi

Departemen Sistem Informasi

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

FINAL PROJECT - ES234849

DEEP LEARNING-DRIVEN DIABETIC RETINOPATHY CLASSIFICATION: FROM OPTIMIZED CNN TO FLASK-BASED DEPLOYMENT

Reyhan Emeraldo

NRP 5026201095

Advisor

Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D.

NIP 1988201812010

Bachelor of Information System Program

Department of Information System

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2025

APPROVAL SHEET

DEEP LEARNING-DRIVEN DIABETIC RETINOPATHY CLASSIFICATION: FROM OPTIMIZED CNN TO FLASK-BASED DEPLOYMENT

FINAL PROJECT

Submitted to fulfill one of the requirements for
obtaining a degree S.Kom at
Undergraduate Study Program of Information Systems Department
of Information Systems
Faculty of Intelligent Electrical and Informatics Technology Institut
Teknologi Sepuluh Nopember

Reyhan Emeraldo

NRP: 5026201095

Approved:

Surabaya, 28 January 2026

Head of Information Systems Departement

LP/P/26/492

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
DEPARTEMEN SISTEM INFORMASI

[illegible]

Prof. Dr. Wiwik Anggraeni, S.Si, M.Kom

NIP. 197601232001122002

APPROVAL SHEET

DEEP LEARNING-DRIVEN DIABETIC RETINOPATHY CLASSIFICATION: FROM OPTIMIZED CNN TO FLASK-BASED DEPLOYMENT

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree S.Kom at
Undergraduate Study Program of Information Systems
Department of Information Systems
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

Reyhan Emeraldo
NRP: 5026201095

Approved by Final Project Examiner Team:

	[illegible]
Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D	(Pembimbing 1)
	[illegible]
Renny Pradina, S.T, M.T	(Penguji 1)
	[illegible]
Prof. Dr. Wiwik Anggraeni, S.Si, M.Kom	(Penguji 2)

SURABAYA
January, 2026

STATEMENT OF ORIGINALITY

The undersigned bellow:

Name of Student / NRP	:	Reyhan Emeraldo / 5026201095
Department	:	S1 Sistem Informasi
Academic Advisor / NIP	:	Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D.. / 1988201812010

Hereby declare that the Final Project with the title “DEEP LEARNING-DRIVEN DIABETIC RETINOPATHY CLASSIFICATION: FROM OPTIMIZED CNN TO FLASK-BASED DEPLOYMENT” is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 20 Januari 2026

Acknowledged, Academic Advisor	Student
[illegible]	METERAI ELEKTRONIK 10000 SEPULUH RIBU RUPIAH TTD Materai 10000 [illegible]
Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D. NIP. 1988201812010	REYHAN EMERALDO NRP.5026201095

ABSTRAK

PENDEKATAN BERBASIS DATA UNTUK MENDETEKSI BURNOUT DALAM KERJA JARAK JAUH MENGGUNAKAN MODEL PEMBELAJARAN MESIN

Nama Mahasiswa / NRP	**:**	**Reyhan Emeraldo / 5026201095**
Departemen	**:**	**Sistem Informasi FTEIC - ITS**
Dosen Pembimbing	**:**	**Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D.**

Abstrak

Retinopati Diabetik (DR) merupakan salah satu penyebab utama kebutaan akibat diabetes baik di negara maju maupun negara berkembang. Identifikasi dini sangatlah penting, namun metode skrining tradisional sering sulit diterapkan karena memerlukan waktu yang lama dan membutuhkan keahlian khusus. Tujuan dari proyek ini adalah merancang dan mengembangkan sistem otomatis yang dapat mendeteksi retinopati diabetik (DR) menggunakan *Convolutional Neural Network* (CNN) tingkat lanjut dan aplikasi web berbasis Flask yang mampu melakukan analisis gambar secara *real-time*.

Model dilatih menggunakan dataset EyePACS, yang memungkinkan proses klasifikasi antara retinopati diabetik (DR) dan non-retinopati diabetik (No DR). Beberapa teknik *preprocessing* diterapkan untuk mengatasi ketidakseimbangan kelas serta meningkatkan performa model. Teknik-teknik tersebut mencakup normalisasi gambar, mengubah ukuran menjadi 150 × 150 piksel, *zooming*, *flipping*, dan *rotating*. Model CNN berhasil mencapai akurasi pengujian sebesar 93% setelah 20 *epochs* pelatihan. Model juga menunjukkan nilai *precision*, *recall*, dan *F1-score* yang baik. Selain itu, model dirancang agar ringan dan cepat.

Flask mempermudah pengguna dalam mengakses model yang telah dilatih. Pengguna dapat mengunggah foto fundus retina melalui antarmuka grafis sederhana dan langsung memperoleh prediksi secara instan. Teknologi ini mampu mendeteksi permasalahan dengan cepat dan akurat. Karya ini memberikan solusi yang sederhana dan dapat diskalakan untuk mendeteksi DR secara dini. Selain itu, proyek ini membuka peluang untuk pengembangan di masa depan, seperti klasifikasi multi-kelas, integrasi *explainable AI*, serta validasi klinis.

Kata kunci: *Retinopati Diabetik (DR), Convolutional Neural Network (CNN), Sistem Deteksi Otomatis, Analisis Gambar Real-Time, Deteksi Dini Penyakit, Aplikasi Web Flask.*

ABSTRACT

DEEP LEARNING-DRIVEN DIABETIC RETINOPATHY CLASSIFICATION: FROM OPTIMIZED CNN TO FLASK-BASED DEPLOYMENT

Student Name / NRP	**:**	**Reyhan Emeraldo / 5026201095**
Department	**:**	**Information System FTEIC - ITS**
Advisor	**:**	**Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D.**

Abstract

Diabetic Retinopathy (DR) is a leading cause of diabetes-induced blindness both in developed and developing countries. Identifying problems early is crucial, but traditional screening methods can be difficult to apply because they are time-consuming and require expertise. The objective of this project is to design and develop an automated system that can detect diabetic retinopathy (DR) within the use of an advanced CNN and a Flask-based web app, which performs real-time image analysis.

The model was trained on the EyePACS dataset, which facilitate the differentiation between diabetic retinopathy (DR) and non-diabetic retinopathy (No DR). Some preprocessing techniques were applied to mitigate the class imbalance and enhance the model's performance. These techniques included normalising images, resizing to 150 x 150 pixels, zooming, flipping, and rotating. The CNN model achieved a test accuracy of 93% after 20 training epochs. It also had confident precision, good recall, and the F1-score. It was going to be light and fast.

Flask made it simple for people to use the trained model. Customers can use a simple graphical interface to upload retinal fundus photos and get predictions right away. The technology was able to quickly and correctly find problems. This work gives us a simple and scalable way to find DR early on. It also sets the stage for future progress, such as multi-class classification, explainable AI integration, and clinical validation.

Keywords: ***Diabetic Retinopathy, Convolutional Neural Network, Automated Diagnosis System, Real-Time Image Analysis, Early Disease Detection, Flask Web Application.***

ACKNOWLEDGMENT

All praise and gratitude are due to Almighty God, Allah SWT, for His abundant grace, mercy, and blessings, which have enabled the author to successfully complete this Final Year Project. Through His guidance and strength, the author was able to overcome various challenges encountered throughout the research and writing process. This Final Year Project is submitted as one of the academic requirements to obtain a Bachelor's Degree in Information Systems under the Double Degree Program between Institut Teknologi Sepuluh Nopember (ITS) and Universiti Teknologi PETRONAS (UTP).

Throughout the completion of this Final Year Project, the author has received invaluable support, guidance, and assistance from many individuals and institutions. Therefore, the author would like to express sincere appreciation and deepest gratitude to the following parties:

1. The Information Systems Department of Institut Teknologi Sepuluh Nopember (ITS) for providing academic support, facilities, and a conducive learning environment that greatly contributed to the author's academic development and research experience.
2. Assistant Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D., as the ITS supervisor, for her continuous guidance, patience, constructive feedback, and valuable insights throughout the preparation and completion of this Final Year Project.
3. Dr. Shashi Bhushan, from Universiti Teknologi PETRONAS (UTP), as the UTP supervisor, for his invaluable supervision, encouragement, and thoughtful guidance from the early stages of the research until the completion of this project. His expertise and suggestions played a crucial role in shaping the direction and quality of this work.
4. All lecturers and academic staff of the Department of Information Systems at ITS, who have shared their knowledge, guidance, and support throughout the author's study period, which laid the foundation for the successful completion of this Final Year Project.
5. The author's beloved family, whose unconditional love, prayers, patience, and moral support have been a constant source of strength and motivation throughout the author's academic journey.
6. Friends and colleagues, especially those from the ITS–UTP Double Degree Program, who provided encouragement, constructive feedback, and companionship during both academic and non-academic moments. Their support and motivation were invaluable during the completion of this Final Year Project.
7. All other parties who cannot be mentioned individually, for their direct or indirect contributions, assistance, and support that helped make the completion of this Final Year Project possible.

Malaysia, 20 August 2025

Author

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DR	Diabetic Retinopathy
FN	False Negative
FP	False Positive
Flask	A Python Micro Web Framework
Grad-CAM	Gradient-weighted Class Activation Mapping
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
ReLU	Rectified Linear Unit
RGB	Red Green Blue (color channel format)
TN	True Negative
TP	True Positive
XAI	Explainable Artificial Intelligence

CHAPTER 1 INTRODUCTION

1.1 Background of Study

Diabetic retinopathy (DR) is a frequent problem for people with diabetes that damages the blood vessels in the retina. It's one of the main reasons why individuals go blind when they don't have to, especially persons who are working age. The problem gets worse as more individuals throughout the world get diabetes. It highlights how vital it is to find strategies to find the condition early and treat it rapidly to improve quality of life and stop eyesight loss [1].

Regular eye exams and trained professionals looking at retinal images by hand are common ways to find DR early. This method shows a lot of problems. First, it requires careful analysis to define small retinal problems like microaneurysms, exudates, and haemorrhages [3]. Second, it is too dependent on the availability and experience of the expert, which can cause delays in some of areas that have limited supporting medical facilities or specialists. Also, the manual interpretation is subjective which could cause differences or even inconsistencies in diagnoses the patient.

Due to these problems and challenges, people want to use the help of artificial intelligence, specifically deep learning methods, to identify DR automatically. The CNN (Convolutional Neural Network) is one of the deep learning models that has presented a lot of promise since CNN can learn complicated hierarchical features from the retinal pictures based on the models without having a lot of manual feature extraction in the architecture. CNN can discover other symptoms of DR in a systematic way, which are faster, reliable, and make it easier for the diagnostic process [5].

The recent improvements in CNN have made the detection of DR more accurate and efficient at the same time. Deep learning frameworks that use DenseNet or residual blocks architecture have shown to be more precise at diagnosing because the better gradient flow and ability to extract more feature that have [4]. Also, new techniques like data augmentation and image preprocessing have been used to improve image become clearer and improve the performance of CNN models. Thus, will make them more resistant to changes in contrast, lighting, and quality that have been common things in clinical retinal images [1].

Even though CNN has improvements, the use of CNN-based systems in real-world practice is still very hard because of things like the difficulty of understanding, how hard they are to compute, and the unbalanced data that is available. Clinicians often need to know how the automated models make decisions, which is why explainable AI is important. Recent search shows the growing need for models that can not only accurately classify DR but also give outputs that are easy to understand and trust by clinicians

To make sure that deep learning-based Diabetic Retinopathy systems can be adapted to real-world clinical settings, it is important to fill in the gaps. So, the goal of this research is to create a CNN-based model that is simple but effective at identifying DR from retinal images. This model will include important preprocessing techniques to help the adaptation in a variety of clinical settings

1.2 Problem Statement

Despite the potential of CNN-based detection systems, there are several challenges that hinder their deployment in real-world clinical environments. The system often depends on balance and high quality dataset, which may not reflect the variability in real applications of clinical images. Many public datasets include noisy, low contrast, or imbalanced images, which can affect model generalizability [4].

Additionally, many advanced deep learning models are intensive computationally and may not be suitable for low-resource deployment. Furthermore, while these models achieve a high performance, their decision-making process often lacks transparency, making the trustworthiness among clinicians limited

There is an urgency for a more efficient, robust, and explainable CNN-based model for DR detection that is applicable to diverse datasets and can be deployed under the constraint of real-world problem.

1.3 Objective

The study objectives are:

1 Develop a CNN (Convolutional Neural Network) model for binary classification of DR using retinal fundus images and deploy it through a Flask-based system

2 Implement preprocessing, including normalization, resizing, and augmentation, to enhance the quality of the image and model robustness.

3 Evaluate the performance of the model using metrics such as the confusion matrix, accuracy, precision, recall, and F1-score.

4 Compare models effectiveness with the recent studies using similar methods and datasets

1.4 Scope of Study

Use 3,662 retinal fundus imaes from EyePACS collection, which was collected through Kaggle Diabetic Retinopathy Detection Competition, this research investigates how to classify whether a person has DR or not. The model is made and constructed using TensorFlow/Keras and consists of three convolutional layers, followed by dense and dropout layers. All of the images undergo preprocessing, which includes normalization and data augmentation techniques like zooming, flipping, and rotating, after being standardized to 150 x 150 pixels. The metrics used to evaluate this model's performance are accuracy, precision, recall, F1-score, and confusion matrix after the test set has been trained for 20 epochs.

Using Flask-based, the trained best model is integrated into a small web application for practical use. Users can upload retinal images and receive real-time predictions via a straightforward interface by using HTML. However, the system does not identify multi-class DR grading or integration into real clinical workflows; it is merely a prototype for local deployment

CHAPTER 2 LITERATURE REVIEW

2.1 Deep Learning in Diabetic Retinopathy Detection

The success of deep learning methods such as Convolutional Neural Networks has advanced the application of artificial intelligence for diabetic retinopathy detection. These models have demonstrated good performance and the ability to detect complex retinal features like exudates, microaneurysms, and haemorrhages from fundus images.

To enhance the classification of diabetic retinopathy, Zago et al. [1] suggested that a CNN-based method can incorporate red lesion localisation. In order to increase the model's interpretability and sensitivity, their approach focused on amplifying and locating lesion regions for classification. Similar to Maity and Chakravorty [5] used CNN to produce an automated DR classification model, emphasizing how it could minimize the screening time and the need of experts.

With the help of OCT (Optical Coherence Tomography), Aggarwal [2] investigated machine learning with retinal pathology. The study highlights the value of artificial intelligence in ophthalmologic diagnosis and shows that machine learning models are transferable across image types, despite the difference between imaging modality and fundus photography

2.2 CNN Architectures and Feature Learning

Convolutional Neural Networks now have deeper models with some advanced feature extraction capabilities evolving from the traditional architectures. A CNN was used by Gaur et al. [3] to classify DR into various levels of severity. Their model was intended to identify contextual and spatial characteristics that are essential for differentiating between proliferative and non-proliferative DR, which was trained on color fundus images

To enhance performance, other research has explored to integrating CNN with othed method. For example, Qomariah et al. [7] evaluated how good the performance of CNN and SVM mehod in the classification of normal and diabetic fundus images. Their research claimed that the use of CNN increased accuracy because it could automatically learn discriminative image features, eliminating the need for manually created features

2.3 Image Preprocessing and Augmentation

To guarantee consistency in model input and eliminate superfluous noise, image preprocessing is crucial. In their comparative study of time series data augmentation techniques for deep learning models, Pedraza et al. [4] highlighted how these techniques improve model generalization and lessen overfitting. Even though their research focused on diabetic neuropathies, retinal image analysis can benefit from the same augmentation techniques (scaling, flipping, and brightness adjustment).

Preprocessing for DR detection frequently consists of image resizing, RGB conversion, contrast enhancement, and Gaussian filter application. By standardizing the dataset, these methods enhance training convergence [6]. In order to compensate for the lack of annotated datasets and class imbalance, data augmentation also makes it possible to train models on artificially diversified samples.

2.4 Explainability and Clinical Trust

Prediction transparency is just as important as accuracy when it comes to the clinical application of deep learning tools. Many AI systems are criticized for being "black boxes," meaning that clinicians cannot see the decision-making processes. P. S. T. et al. [11] addressed this by proposing explainable deep learning models that highlight image regions that contribute to the final classification by creating attention maps. Ophthalmologists can better comprehend and validate the model's predictions with the aid of these visual explanations.

Similarly, Surabhi and Prasad [8] improved clinician trust and aided junior medical staff training by incorporating explainable decision support into their DR classification system. Thus, Grad-CAM and related methods are now a crucial part of AI-powered medical applications.

Apart from model transparency, user-facing systems can be crucial in connecting clinical workflows with AI predictions. For instance, incorporating CNN models into straightforward web-based interfaces enhances trust by providing clear channels for interaction with the model and enabling real-time predictions and visual result display.

2.5 Efficient and Lightweight Models

Deeper CNNs make predictions more accurate, but they can be hard on computers. This could limit how they can be used, especially in edge or mobile settings. Elzennary et al. [10] suggested a lightweight CNN model for detecting DR that is good for real-time inference and has a lower computational load. They showed that optimized architectures with fewer parameters could still perform well in competition, which made them good for use in portable diagnostic tools.

2.6 Model Evaluation

To fully assess how well a diabetic retinopathy classification model works, you need to look at more than just its accuracy. To check how well a model works, find bias, and help with further optimization, people usually use a mix of quantitative metrics and visualization techniques.

2.6.1 Accuracy

The accuracy of a prediction is the number of correct predictions divided by the total number of predictions. It is often used for classification problems, but it may not show how well the model works when there is class imbalance. For example, a model might seem very accurate at finding diabetic retinopathy (DR) by often predicting the dominant class (non-DR), which could mean that it misses real DR cases. Still, accuracy is a quick and easy way to tell how well you're doing overall [6].

2.6.2 Confusion Matrix

A confusion matrix shows the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) to give a better picture of how well a model works. This helps find certain kinds of misclassifications and see how well the model separates classes. Zago et al. [1] showed that confusion matrices are very important for testing how well a model can tell the difference between DR-positive cases and normal ones, especially when early-stage DR is hard to find.

2.6.3 Precision, Recall, and F1-Score

Precision shows how many of the predicted positive cases are actually true. This shows how well the model can avoid false positives. Recall shows how sensitive the model is by showing how well it finds real positive cases. The F1-score gives an evaluation by combining recall and precision into one new metric. This will be helpful when classes are not balanced. Many previous DR studies use these metrics, such as the study of Gaur et al. [3], which used the F1-score to determine the most reliable CNN architecture.

2.6.4 Training and Validation Loss/Accuracy Curves

Plotting the training and validation accuracy/loss curves over time will shows if the models is overfitting or underfitting. Bad generalization can be determined by a big difference between training and validation accuracy. Surabhi and Prasad [8] tells how important to see loss in order to choose early stopping points and make the most of training time to keep the model in good performance and neglect performance loss.

Study	**Model Type**	**Focus**	**Dataset**	**Accuracy**
Zago et al. (2020)	Deep network patch-based approach (two CNN models) for lesion localization	Detecting early signs of DR; automatically localizing lesion regions (red lesions)	Trained on DIARETDB1; tested on several databases (including Messidor)	AUC: 0.912
Gaur et al. (2025)	CNN-based method utilizing DenseNet169 architecture	Diabetes prediction using retinal images; DR prediction; Classifying retinal images into five DR stages (0-4)	APTOS 2019 blindness detection dataset (approx. 13,000 retinal images)	Over 82%

Padhmapriya et al. (2025)	Pre-trained Deep Learning Models (AlexNet, GoogLeNet, ResNet, VGGNet) with transfer learning and fine-tuning	Diabetic Retinopathy (DR) detection; Clinical decision support; Automated DR screening	Balanced dataset of 3,662 retinal scan images	Not explicitly stated in abstract
Elzennary et al. (2020)	Computationally efficient deep learning CNN based on DenseNet-121 architecture with transfer learning	Automated DR diagnosis; Building reliable and computationally efficient deep learning model; Detecting severity of the disease	Commonly used labeled retinal images dataset (from Kaggle)	95.64% (validation accuracy)
Bilal et al. (2021)	Novel Hybrid Approach (combining SVM, KNN, Binary Trees with preprocessing and feature extraction)	Prior DR detection and classification for disease grading	Multiple severities of disease grading databases	98.06% (Sensitivity: 83.67%, Specificity: 100%)
A Raghu Vira Pratap et al. (2024)	Integrated IDNet and Residual Networks (ResNet); includes data augmentation	Efficient Diabetic Retinopathy Grading; Diagnosing DR	Publicly accessible datasets such as Messidor, Messidor-2, DRISHTI-GS,	ROC AUC: 0.94 (Precision: 0.924, Recall: 0.955, F1-score: 0.931)

			and a Retinal Dataset from GitHub	
Pedraza et al. (2024)	Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNN) with various data augmentation techniques	Empirical comparative analysis of data augmentation techniques for diabetic neuropathy detection using time series analysis	Preprocessed dataset (postural time series data)	Up to 100% (with Jittering for MLP & CNN)
Aggarwal (2019)	Deep convolutional neural network (Retrained VGG16)	Examining the effect of data augmentation on performance for analyzing ophthalmological diseases (AMD, CSR, DR, MH) using OCT images	Optical Coherence Tomography Image Database	AUC: 0.97 (with data augmentation)

Table 2-1 Comparative Summary of Recent Deep Learning Approach

CHAPTER 3 METHODOLOGY

3.1 Overview

This chapter will explain how a Convolutional Neural Network (CNN)-based system was made to classify diabetic retinopathy in retinal fundus images. The method is designed to achieve high classification performance while keeping it usable and lightweight in an environment with limited computational resources. This chapter goes into detail about the dataset's features, the image preprocessing methods used, the architecture of the CNN model, the training setup, and the evaluation metrics used to measure how well the model worked.

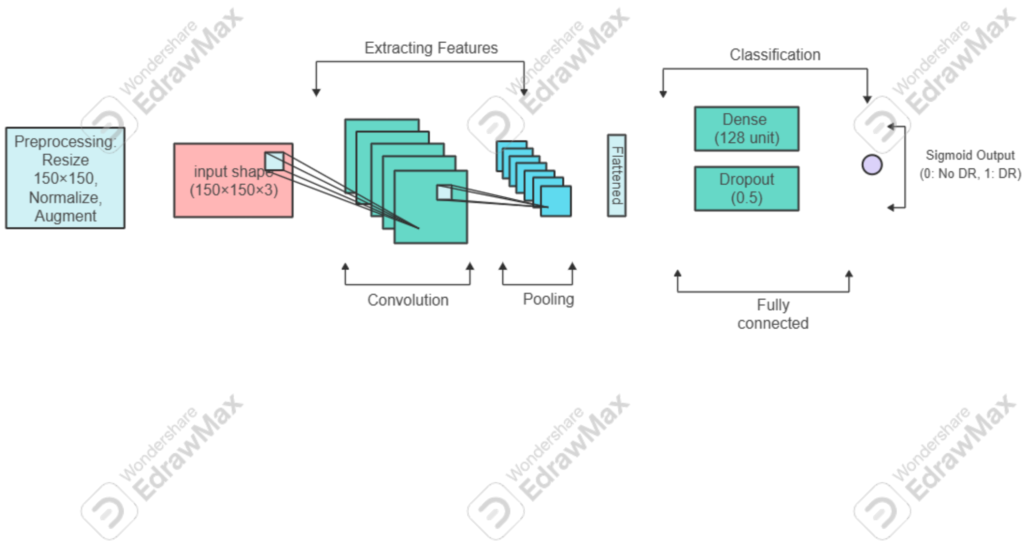


Figure 3-1CNN Architecture and Image Preprocessing

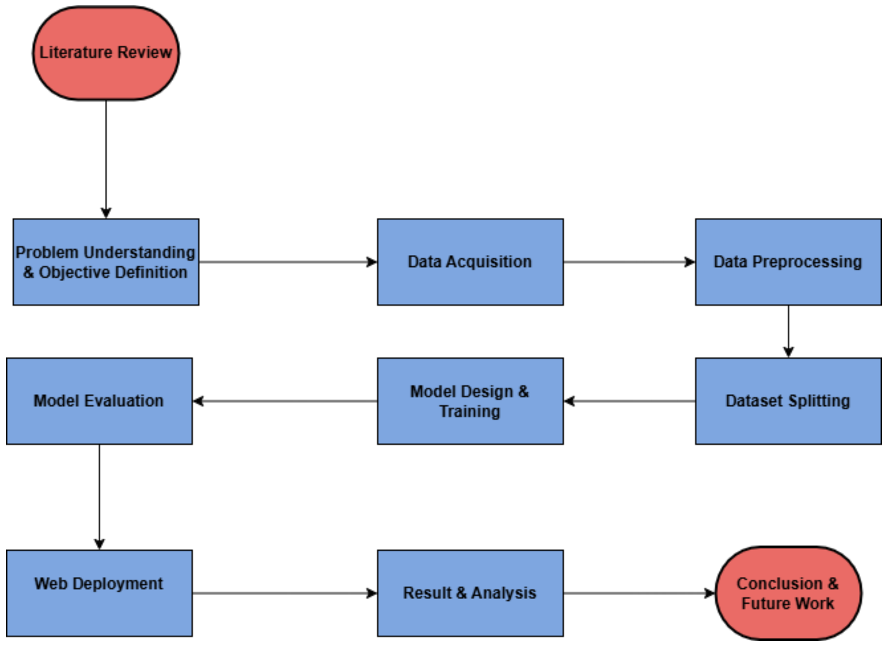


Figure 3-2 Research Flowchart

3.2 Dataset Description

The original EyePACS dataset has more than 35,000 labeled retinal images, but this study only used 3,662 of them. The choice was made to keep the class distribution even (1,875 images with DR and 1,805 without DR) while still being able to work within the limits of the development environment (Google Colab with limited GPU and storage space). This smaller but still representative sample made it possible to train and test the model quickly without losing the study's clinical relevance.

Originally, the dataset includes five severity classes:

- **0** – No DR
- **1** – Mild
- **2** – Moderate
- **3** – Severe
- **4** – Proliferative DR

However, for the purpose of this study, the classification task was simplified into a binary classification problem:

- **Class 0**: No DR

- **Class 1**: DR present (combining Classes 1–4)

This simplification makes it easier to deal with differences between classes and lets the model focus on finding out if there is a disease or not, which is an important first step in screening applications. There is a big class imbalance in the data, even though there are a lot of it (about 35,000 training images). Too many images are labelled as "No DR." To improve generalization, augmentation techniques must be used because of this imbalance.

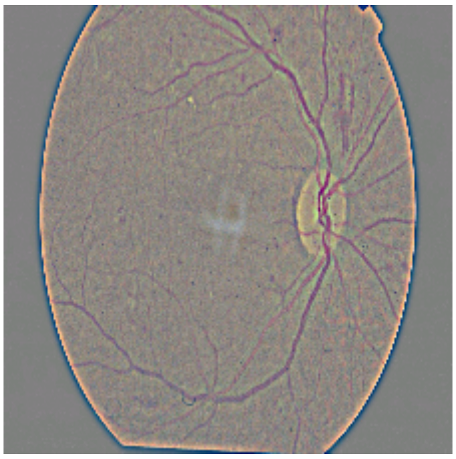


Figure 3-3 Example of Retinal Fundus Image of Diabetic Retinopathy

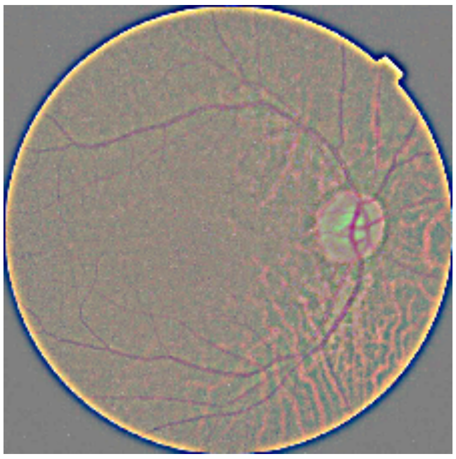
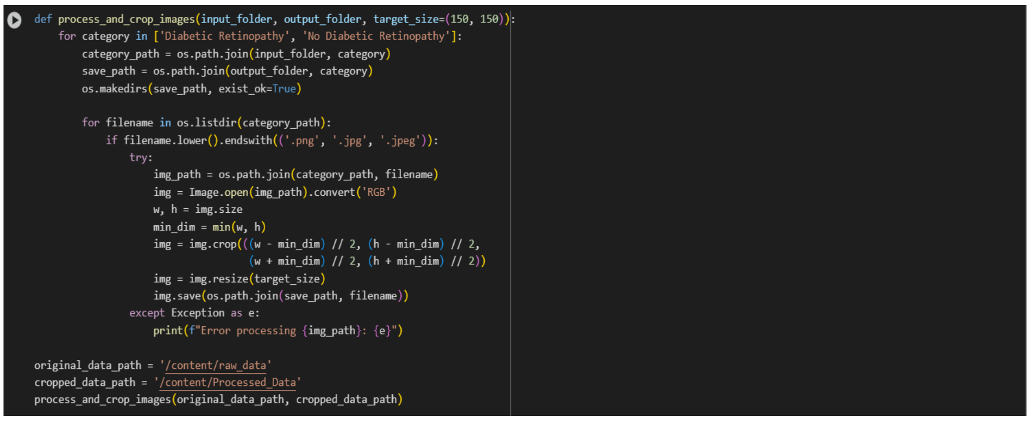


Figure 3-4 Example of Normal Retinal Fundus Image

3.3 Data Preprocessing

Because the original images had problems with brightness, contrast, and size, a lot of preprocessing was done to make sure that the data was consistent and of high quality for the CNN model.



```python
def process_and_crop_images(input_folder, output_folder, target_size=(150, 150)):
    for category in ['Diabetic Retinopathy', 'No Diabetic Retinopathy']:
        category_path = os.path.join(input_folder, category)
        save_path = os.path.join(output_folder, category)
        os.makedirs(save_path, exist_ok=True)

        for filename in os.listdir(category_path):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                try:
                    img_path = os.path.join(category_path, filename)
                    img = Image.open(img_path).convert('RGB')
                    w, h = img.size
                    min_dim = min(w, h)
                    img = img.crop(((w - min_dim) // 2, (h - min_dim) // 2,
                                    (w + min_dim) // 2, (h + min_dim) // 2))
                    img = img.resize(target_size)
                    img.save(os.path.join(save_path, filename))
                except Exception as e:
                    print(f"Error processing {img_path}: {e}")

original_data_path = '/content/raw_data'
cropped_data_path = '/content/Processed_Data'
process_and_crop_images(original_data_path, cropped_data_path)
```

Figure 3-5 Phyton Code for Image Preprocessing and Resizing

3.3.1 Image Cropping and Resizing

Different aspect ratios were shown in the raw images, and they often had extra black borders or artifacts. To make the dataset consistent:

- All images were cropped to a square shape by removing excess width or height, centering the retina.
- They were then resized to 150×150 pixels, balancing computational efficiency with sufficient detail retention for lesion recognition.

This step significantly reduces memory consumption and enables faster training without a notable loss in image quality.

3.3.2 Color Chanel Normalization

All images were:

- Converted to RGB format, ensuring consistency in input channels.
- Normalized by dividing pixel values by 255 to scale them into the range [0, 1].

This normalization helps stabilize and accelerate model convergence during training.

3.3.3 Data Augmentation

Because the dataset was uneven and had different acquisition conditions, ImageDataGenerator was used to add to it on the fly. Some of the methods used were:

- Rotation (±15°): To simulate varied head tilts
- Zoom (up to 20%): To mimic different focus levels
- Width/Height Shift (±10%): To create translational invariance
- Shear Transformation
- Horizontal Flipping: Useful for symmetric anatomical structures

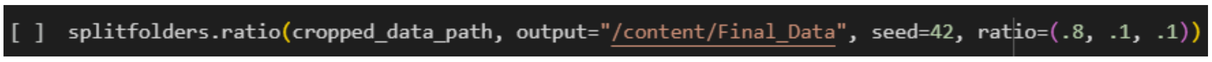
These changes do a good job of making the dataset more diverse, lowering the risk of overfitting, and making the model more robust.

3.3.4 Dataset Splitting

The preprocessed dataset was divided using splitfolders.ratio() with the following ratio:

- 80% training set
- 10% validation set
- 10% test set

Each subset maintained class distribution to prevent bias in learning or evaluation.



```
[ ]  splitfolders.ratio(cropped_data_path, output="/content/Final_Data", seed=42, ratio=(.8, .1, .1))
```

Figure 3-6 Code Snippet for Splitting Dataset

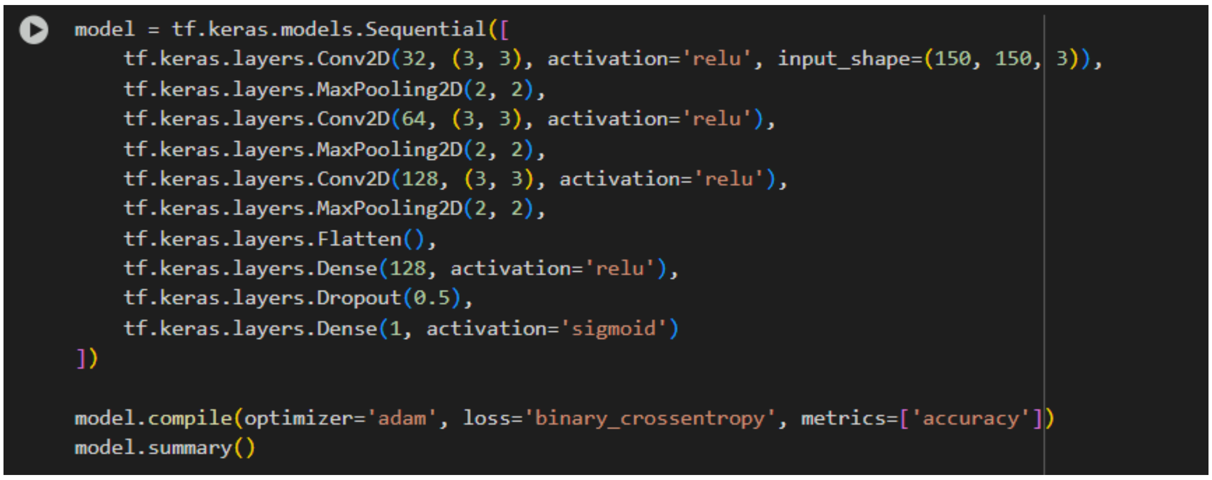
3.4 Model Architecture

To ensure data quality and consistency, the following preprocessing steps are applied based on the provided code files:

TensorFlow/Keras (Sequential API) was used to implement the CNN model. Lightweight and moderately deep, the architecture is appropriate for deployment in settings with constrained computational power. The following layers make up the design, which was influenced by conventional CNN pipelines used for image classification tasks:

Layer	Details
Conv2D (1st layer)	32 filters, 3×3 kernel, ReLU activation, input shape = (150, 150, 3)
MaxPooling2D	2×2 pool size
Conv2D (2nd layer)	64 filters, 3×3 kernel, ReLU activation
MaxPooling2D	2×2 pool size
Conv2D (3rd layer)	128 filters, 3×3 kernel, ReLU activation
MaxPooling2D	2×2 pool size
Flatten	Converts 3D output to 1D for dense layers
Dense (Fully Connected)	128 units, ReLU activation
Dropout	0.5 rate — randomly disables 50% of neurons during training
Dense (Output Layer)	1 unit, Sigmoid activation — outputs binary probability (DR vs. No DR)

Table 3-1 Summary of the CNN Architecture Used



```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

Figure 3-7 Code Snippet Defining the CNN Architecture

3.5 Model Training Configuration

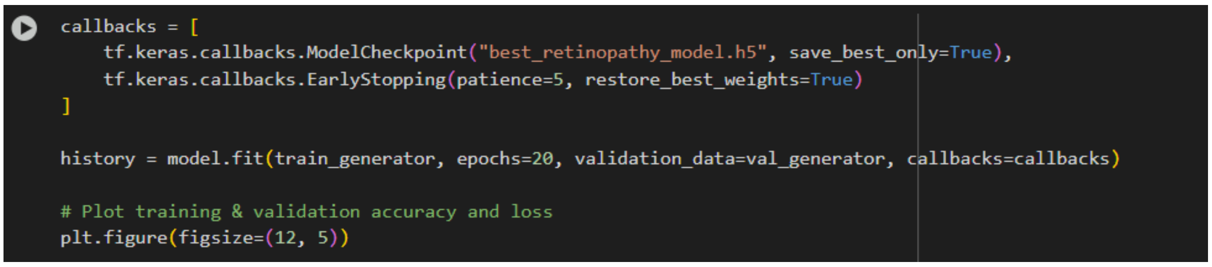
The following parameters were used to put together and train the model:

- Loss Function: binary_crossentropy appropriate for binary classification
- Optimizer: Adam adaptive learning rate and widely used for image models
- Metrics: accuracy

The training was carried out for 20 epochs with the following callbacks:

- EarlyStopping: Halts training when validation loss stops improving (patience = 5)
- ModelCheckpoint: Saves the model with the best validation accuracy

Training took place on Google Colab, which has a GPU-enabled environment (usually a Tesla T4 with 12 GB VRAM). This made the training time shorter and the model work better.



```
callbacks = [
    tf.keras.callbacks.ModelCheckpoint("best_retinopathy_model.h5", save_best_only=True),
    tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)
]

history = model.fit(train_generator, epochs=20, validation_data=val_generator, callbacks=callbacks)

# Plot training & validation accuracy and loss
plt.figure(figsize=(12, 5))
```

Figure 3-8 Code Snippet for Model Training

3.6 Evaluation Metrix

3.6.1 Accuracy

Accuracy was used as a baseline measure to see how many of the predicted outcomes were correct. But because there is an imbalance in the classes, accuracy alone is not enough to show how well the model works.

3.6.2 Confusion Matrix

The confusion matrix provides a breakdown of prediction outcomes:

- True Positives (TP): Correctly detected DR cases
- True Negatives (TN): Correctly identified non-DR cases

- False Positives (FP): Normal cases incorrectly classified as DR
- False Negatives (FN): Missed DR cases

This matrix was plotted to visualize and analyze misclassification trends.

3.6.3 Precision, Recall, and F1-Score

To supplement accuracy:

- Precision checks how many of the predicted DR cases were correct.
- Recall (or Sensitivity) checks how many real DR cases were found.
- The F1-Score balances both metrics and gives you a harmonic mean, which is great for datasets that aren't balanced.

We used sklearn.metrics.classification_report to figure out these numbers.

3.6.4 Training and Validation Curves

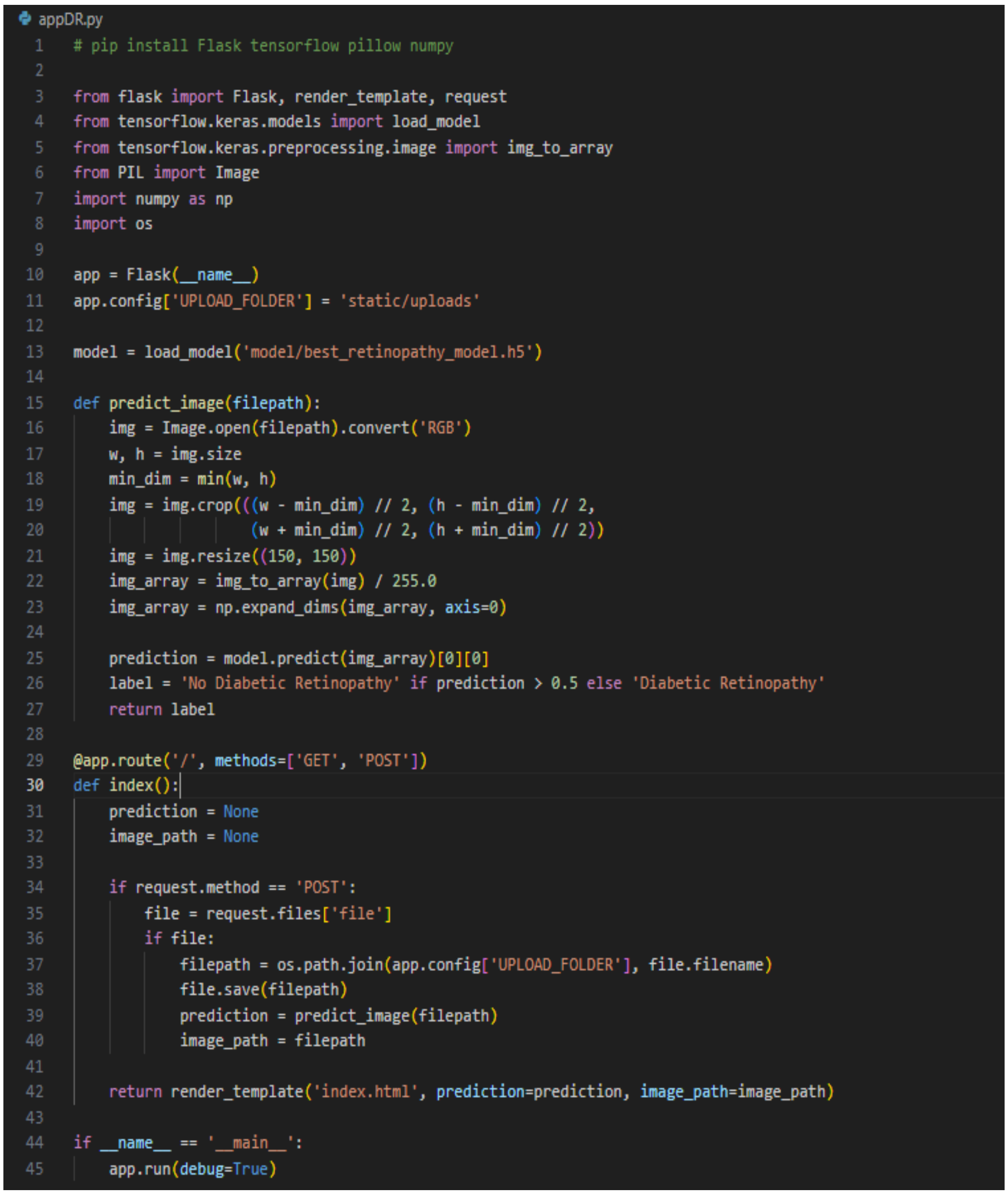
Line plots were generated to visualize:

- The accuracy of training and validation.
- The loss of training and validation.

These curves help find overfitting (when the validation loss goes up while the training loss goes down) and make sure the model works well on data it has not seen before.

3.7 System Implementation

We used the Flask web framework to put the trained CNN model into a lightweight web app so that it could make predictions in real time and be more useful in real life. The system is a front-end interface that lets you upload retinal fundus images and get an immediate binary classification output (DR or No DR).


appDR.py

```python
# pip install Flask tensorflow pillow numpy

from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array
from PIL import Image
import numpy as np
import os

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'static/uploads'

model = load_model('model/best_retinopathy_model.h5')

def predict_image(filepath):
    img = Image.open(filepath).convert('RGB')
    w, h = img.size
    min_dim = min(w, h)
    img = img.crop(((w - min_dim) // 2, (h - min_dim) // 2,
                    (w + min_dim) // 2, (h + min_dim) // 2))
    img = img.resize((150, 150))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)[0][0]
    label = 'No Diabetic Retinopathy' if prediction > 0.5 else 'Diabetic Retinopathy'
    return label

@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    image_path = None

    if request.method == 'POST':
        file = request.files['file']
        if file:
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
            file.save(filepath)
            prediction = predict_image(filepath)
            image_path = filepath

    return render_template('index.html', prediction=prediction, image_path=image_path)

if __name__ == '__main__':
    app.run(debug=True)
```

Figure 3-9 Code Snippet for Model Compilation, Training and Callback Configuration

3.7.1 System Architecture

The web-based system consists of the following components:

Component	Description
Frontend	Created with HTML and CSS. Has a form for uploading images and a styled way to show the results
Backend	Made with Flask. Takes care of loading models, preprocessing images, and making predictions
Model	The trained CNN (.h5 format) was loaded at runtime for inference in real time

Output	Shows either "Diabetic Retinopathy" or "No Diabetic Retinopathy" along with a preview of the uploaded image

Table 3-2 Summary of Web-Based Diabetic Retinopathy Detection System Components

3.7.2 Image Processing Pipeline

Uploaded images are:

- Changed to RGB format
- Cropped to a square and resized to 150×150 pixels
- Normalized to a [0–1] pixel range

This make sure that the input is always the same before the prediction is made with the same CNN model that was trained before

3.7.3 Deployment Context

The app is currently running on a local Flask server, which is good for testing and showing off. In the future, the system can be expanded to work on cloud platforms like Render and Heroku, or it can be turned into a mobile app with REST API endpoints.

CHAPTER 4 RESULT & DISCUSSION

4.1 Overview

This chapter talks about what happened when a Convolutional Neural Network (CNN) model was used to classify diabetic retinopathy (DR) images as either having or not having the disease. We used metrics like accuracy, precision, recall, F1-score, and the confusion matrix to measure how well the model worked. Also, a graphical user interface (GUI) was made to show how the model can make predictions in real time. The results are also compared to those of other studies that looked at similar things.

4.2 Training Performance

We trained the model for 20 epochs on Google Colab, which has a GPU. We kept track of and plotted the accuracy and loss for training and validation (Figures 10).

- The accuracy of the training slowly get better, reaching about 91%.
- The accuracy of the validation stayed between 91% and 93%
- The training and validation loss both went down over epochs, with only small difference, which means the model wasn't overfitting

These trends show that the model can generalise well and learn effectively

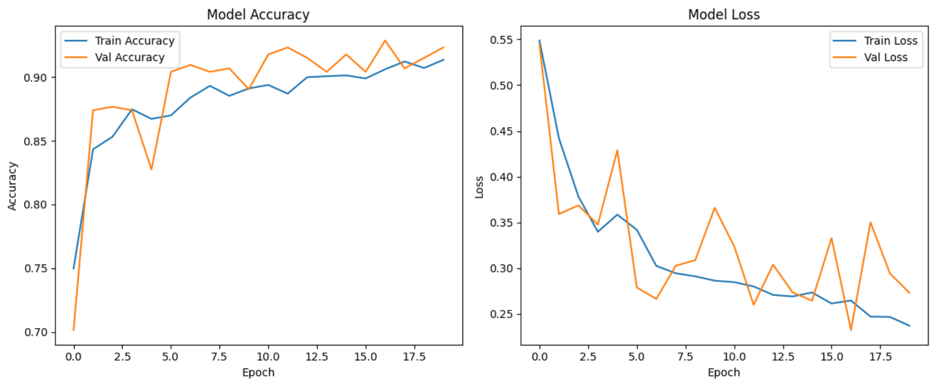


Figure 4-1 Training and Validation Accuracy and Loss Curves Across Epochs

4.3 Test Set Distribution

The final dataset was divided into three parts: training (80%), validation (10%), and testing (10%). There were 368 images in the test set, and they were divided into the following classes:

Class	Number of Images
Diabetic Retinopathy	187
No Diabetic Retinopathy	181
Total	**386**

Table 4-1 Distribution of Test Dataset Samples by Class

This distribution is nearly balanced, allowing for fair evaluation across both classes.

4.4 Confusion Matrix Analysis

The model's predictions on the test set were evaluated using a confusion matrix, shown in Table 5

Actual \ Predicted	Diabetic Retinopathy	No Diabetic Retinopathy
Diabetic Retinopathy	169 (True Positive)	18 (False Negative)
No Diabetic Retinopathy	7 (False Positive)	174 (True Negative)

Table 4-2 Confusion Matrix of CNN Model Predictions

This demonstrates strong diagnostic capability for both classes:

- High true positive rate for DR (168/187)
- High true negative rate for non-DR (174/181)

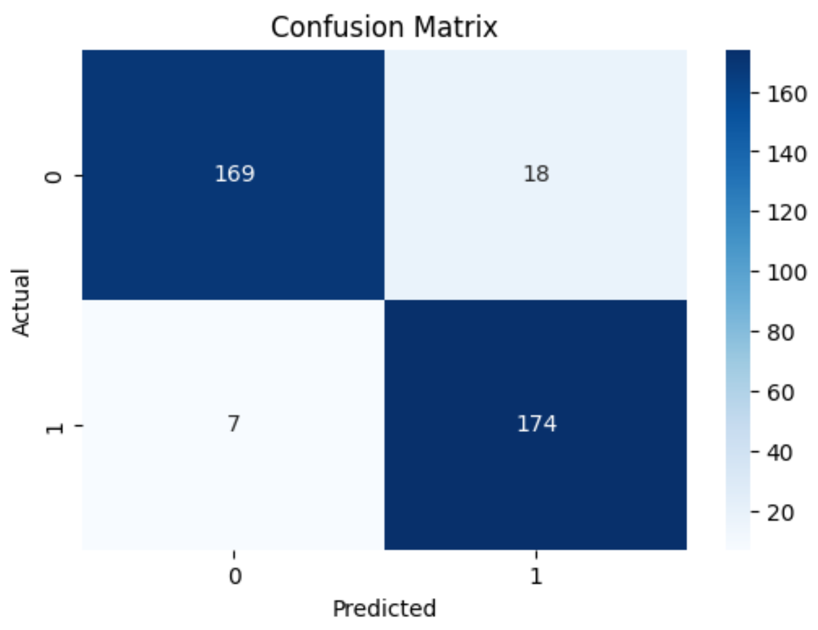


Figure 4-2 Confusion Matrix of CNN Model on the Test Dataset

4.5 Classification Metrics

The detailed evaluation report from the classification_report() function is summarized below:

Class	**Precision**	**Recall**	**F1-score**	**Support**
Diabetic Retinopathy	0.96	0.90	0.93	187
No Diabetic Retinopathy	0.91	0.96	0.93	181
Accuracy			**0.93**	**368**
Macro Average	0.93	0.93	0.93	
Weighted Average	0.93	0.93	0.93	

Table 4-3 Classification Report of CNN Model

4.6 Comparison with Previous Studies

The model performance is compared with previous works in Table 4.2:

Study	Dataset	Accuracy	Method Description
Zago et al. (2020)	EyePACS	85.3%	Red lesion localization
Surabhi S (2023)	APTOS 2019 Blindness Detection dataset	92% (DenseNet-201)	Convolutional Neural Networks (CNN) and DenseNet-201
Gaur et al. (2025)	APTOS 2019 blindness detection dataset (approx. 13,000 retinal images)	Over 82%	CNN-based method utilizing DenseNet169 architecture
This Study	**EyePACS (Kaggle)**	**93.0%**	**3-layer CNN + Dropout + GUI**

Table 4-4 Performance Comparison of the Proposed Model with Existing Studies

The proposed model works better than similar lightweight CNN architectures while keeping a simple structure, which makes it good for use in places with few resources.

4.7 GUI Integration and Usability

We made a Flask-based GUI to simulate real-time screening to make it easier to use. With this interface, users can upload fundus images and get classification results right away.

Features**:**

- Upload image
- Predict DR presence
- Display predicted label and uploaded image
- Intuitive design for non-technical users

This component validates the system's practical applicability, particularly in telemedicine or mobile healthcare settings.

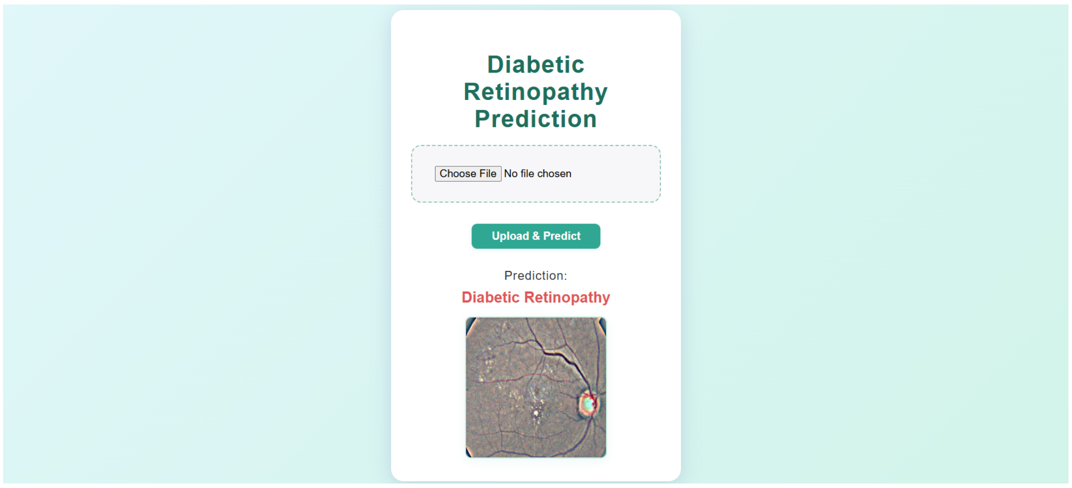


Figure 4-3 Screenshot of Flask-Based Web Interface

4.8 Discussion

The model was able to generalise well, getting 93% of the test right and an F1 score of 0.93 across both classes. It is perfect for use in screening systems with limited resources due to its simplicity and good performance.

Key Points:

- Minimal architecture yields competitive results
- Balanced class performance minimizes false diagnoses
- GUI interface bridges model usability and real-world impact

The models achieve remarkable performance, especially considering the lighting of the images. It is also ready to be used in real-world scenarios for future clinical testing or integration into a real web-based based.

CHAPTER 5 CONCLUSION & RECOMMENDATION

5.1 Conclusion

The study was able to produce and test a Convolutional Neural Network (CNN) model that can identify diabetic retinopathy (DR) early and classify it into two groups using the fundus image of the retina. The model was trained on the EyePACS dataset, which is available publicly on Kaggle. The wide range of retinal images shows that the dataset it can be representative clinically. The preprocessing techniques improve the model capability to handle and generalise changes in the real world retinal images.

The model achieves an overall accuracy of 93% on the test dataset. The other metrics, such as recall, precision, and F1-score values, were all around 0.93 for both the DR-positive and DR-negative classes. This shows that the model is trustworthy at identifying diabetic retinopathy, which means useful as a practical screening tool.

The model's confusion matrix analysis showed that it worked well, with low rates of false negatives and false positives, which is a very important aspect for clinical screening or diagnosing. The proposed CNN architecture shows a better performance than the previous study while still being computationally efficient, making it suitable for general use with limited resources

The deployment of a trained Convolutional Neural Network model into a Flask-based web application showed the usefulness of real-time use. The graphical user interface (GUI) was made to help users in real-time DR prediction.

5.2 Recommendation

1. Multi-class Classification:
   Build models that can create a multi-class grading system for the severity of diabetic retinopathy with binary classification (mild, moderate, severe, and proliferative). This would help medical to get more and detailed information for decision making and diagnosing.

2. Explainability and Interpretability:
   Include an explainable artificial intelligence (XAI) like Grad-CAM or attention maps to make the model more transparent to help medical experts understand and trust AI-driven decisions.

3. Clinical Validation:
   Testing with the real-world clinical datasets from local hospitals or clinics to make the model more reliable and credible for diabetic retinopathy detection.

4. Deployment in Resource-limited Settings:
   Making the model work better on mobile devices or edge computing environments (like Raspberry Pi) could make it more available in rural and underserved areas, giving people a useful diagnostic tool where there aren't many ophthalmologists.

5. User and Expert Feedback:
   Future versions should include structured feedback from ophthalmologists and healthcare providers to make the model and interface even better, making sure they fit with clinical workflows and user needs.

In conclusion, this study is a big step forward in the automated detection of diabetic retinopathy. It shows how machine learning and real-world use can make healthcare better. Continued research and development in the suggested areas could greatly improve the early diagnosis, treatment, and management of diabetic retinopathy around the world.

5.3 Limitations

Even though the research got some good results, it had a lot of problems:

- There were only 3,662 retinal images in the dataset, which is a small part of the full EyePACS dataset. This small amount of data may make the model less generalisable and robust when used in more clinical settings.
- The model only looked at binary classification (DR vs. No DR), which is fine for early screening but doesn't help with the different stages of diabetic retinopathy severity that are needed for a full medical diagnosis.
- The Flask-based web app can only be used in a limited area right now. It wasn't tested on real clinical data or used in real healthcare systems.

- The model doesn't have features that make it easy to understand, like Grad-CAM or attention maps, which are important for getting doctors to use it and for building trust among them.

To improve generalisation, future research should include a bigger and more varied image dataset to get around these problems. It is suggested that the system be expanded to include multi-class classification, explainable AI (XAI) methods, and clinical validation using real-world hospital data. Also, putting the system on cloud platforms or mobile devices will make it easier to use in real life and in telemedicine and underserved areas.

REFERENCES

[1] G. T. Zago et al., "Diabetic Retinopathy Detection Using Red Lesion Localization and Convolutional Neural Networks," *Computers in Biology and Medicine*, vol. 116, p. 103545, 2020. doi: 10.1016/j.compbiomed.2019.103537

[2] P. Aggarwal, "Machine learning of retinal pathology in optical coherence tomography images," *Journal of Medical Artificial Intelligence*, vol. 2, 2019. [Online]. Available: https://jmai.amegroups.org/article/view/5145

[3] S. Gaur, A. Kandwal, and B. Pandey, "Detection of diabetic retinopathy and classification of its stages by using convolutional neural network," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 37, no. 2, pp. 1284–1293, 2025. doi: 10.11591/ijeecs.v37.i2.pp1284-1293.

[4] N. Pedraza, C. Villegas, D. C. Aqueveque, and R. Das, "A Comparative Analysis of Time Series Data Augmentation Methods in the Identification of Diabetic Neuropathies Based on Deep Learning Algorithms," in *Proc. 2024 IEEE Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (SCCC)*, pp. 1–8, 2024. doi: 10.1109/SCCC63879.2024.10767645.

[5] P. Maity and C. Chakravorty, "AI Based Automated Detection & Classification of Diabetic Retinopathy," *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, Bangalore, India, 2023, pp. 1-6, doi: 10.1109/CSITSS60515.2023.10334199.

[6] M. F. Mostafa, H. Khan, F. Farhana, and M. A. H. Miah, "Application of Deep Learning Framework for Early Prediction of Diabetic Retinopathy," *AppliedMath*, vol. 5, no. 1, p. 11, 2025, doi: 10.3390/appliedmath5010011.

[7] D. U. N. Qomariah, H. Tjandrasa and C. Fatichah, "Classification of Diabetic Retinopathy and Normal Retinal Images using CNN and SVM," 2019 12th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 2019, pp. 152-157, doi: 10.1109/ICTS.2019.8850940.

[8] S. Surabhi and R. Prasad, "Diabetic Retinopathy Classification using Deep Learning Techniques," *2023 7th International Conference on Emerging Approaches in Science, Computing and Technology (EASCT)*, [City, Country of conference - not provided in source], 2023, pp. 1-6, doi: 10.1109/EASCT59475.2023.10392721.

[9] A. Bilal, G. Sun, Y. Li, S. Mazhar, and A. Q. Khan, "Diabetic Retinopathy Detection and Classification Using Mixed Models for a Disease Grading Database," *IEEE Access*, vol. 9, pp. 23544-23553, 2021, doi: 10.1109/ACCESS.2021.3056186.

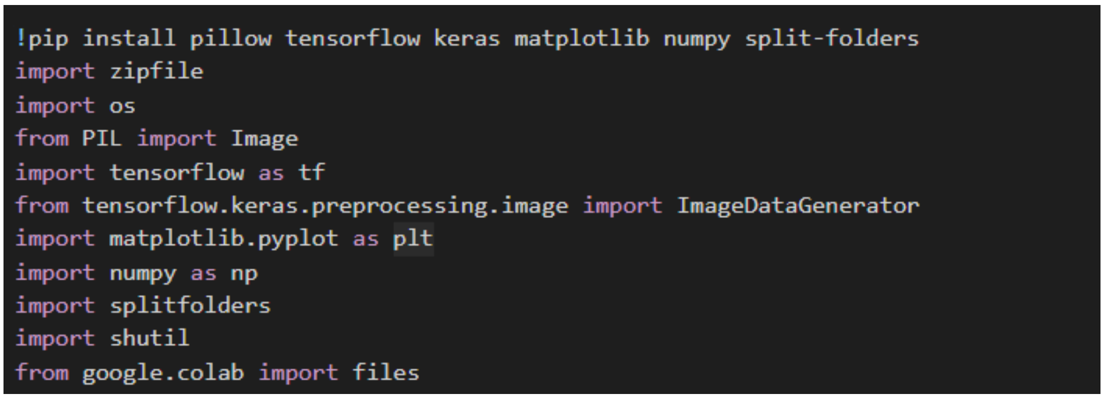
[10] A. Elzennary, M. Soliman, and M. Ibrahim, "Early Deep Detection for Diabetic Retinopathy," *2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, Marrakech, Morocco, 2020, pp. 1-5, doi: 10.1109/ISAECT50560.2020.9523650.

[11] P. S. T, S. M, K. S. A, and P. J. B, "Explainable Deep Learning Models for Clinical Decision Support in Diabetic Retinopathy Detection," *2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp. 1-6, doi: 10.1109/IATMSI64286.2025.10984647.

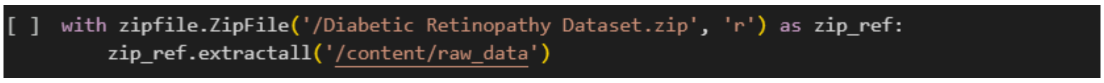
[12] A. R. V. Pratap, A. V. S. S. Nanditha, T. Laahiri, and M. Nurulla, "Integrating IDNET and Residual Blocks for Efficient Diabetic Retinopathy Grading," *2024 5th IEEE Global Conference for Advancement in Technology (GCAT)*, Bangalore, India, 2024, pp. 1-6, doi: 10.1109/GCAT62922.2024.10923934.

[13] E. Dugas, J. Jared, J. Jorge, and W. Cukierski, "Diabetic Retinopathy Detection [Data set]," Kaggle, 2015. [Online]. Available: https://www.kaggle.com/competitions/diabetic-retinopathy-detection

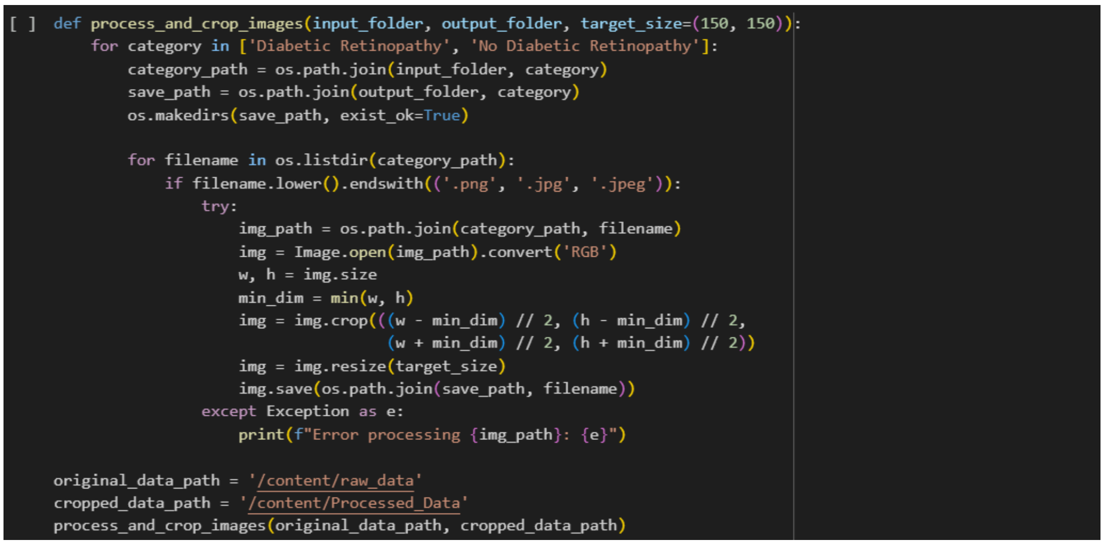
APPENDIX



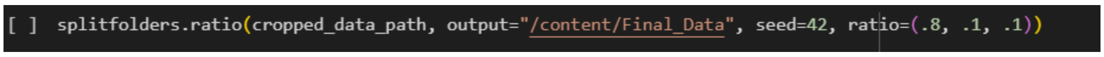
```
!pip install pillow tensorflow keras matplotlib numpy split-folders
import zipfile
import os
from PIL import Image
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
import splitfolders
import shutil
from google.colab import files
```



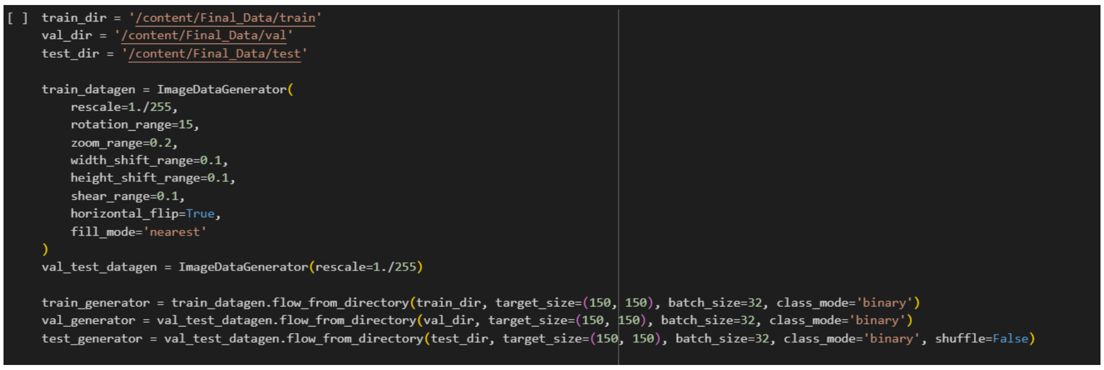
```
[ ]  with zipfile.ZipFile('/Diabetic Retinopathy Dataset.zip', 'r') as zip_ref:
         zip_ref.extractall('/content/raw_data')
```



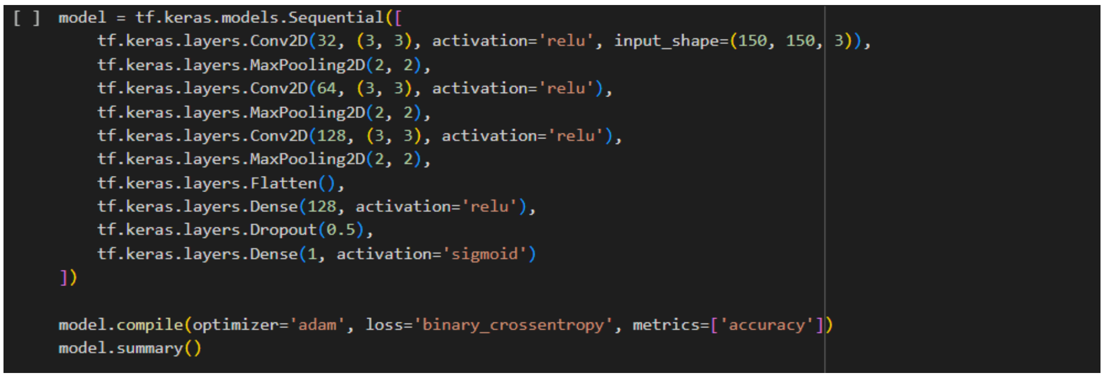
```
[ ]  def process_and_crop_images(input_folder, output_folder, target_size=(150, 150)):
         for category in ['Diabetic Retinopathy', 'No Diabetic Retinopathy']:
             category_path = os.path.join(input_folder, category)
             save_path = os.path.join(output_folder, category)
             os.makedirs(save_path, exist_ok=True)

             for filename in os.listdir(category_path):
                 if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                     try:
                         img_path = os.path.join(category_path, filename)
                         img = Image.open(img_path).convert('RGB')
                         w, h = img.size
                         min_dim = min(w, h)
                         img = img.crop(((w - min_dim) // 2, (h - min_dim) // 2,
                                         (w + min_dim) // 2, (h + min_dim) // 2))
                         img = img.resize(target_size)
                         img.save(os.path.join(save_path, filename))
                     except Exception as e:
                         print(f"Error processing {img_path}: {e}")

     original_data_path = '/content/raw_data'
     cropped_data_path = '/content/Processed_Data'
     process_and_crop_images(original_data_path, cropped_data_path)
```



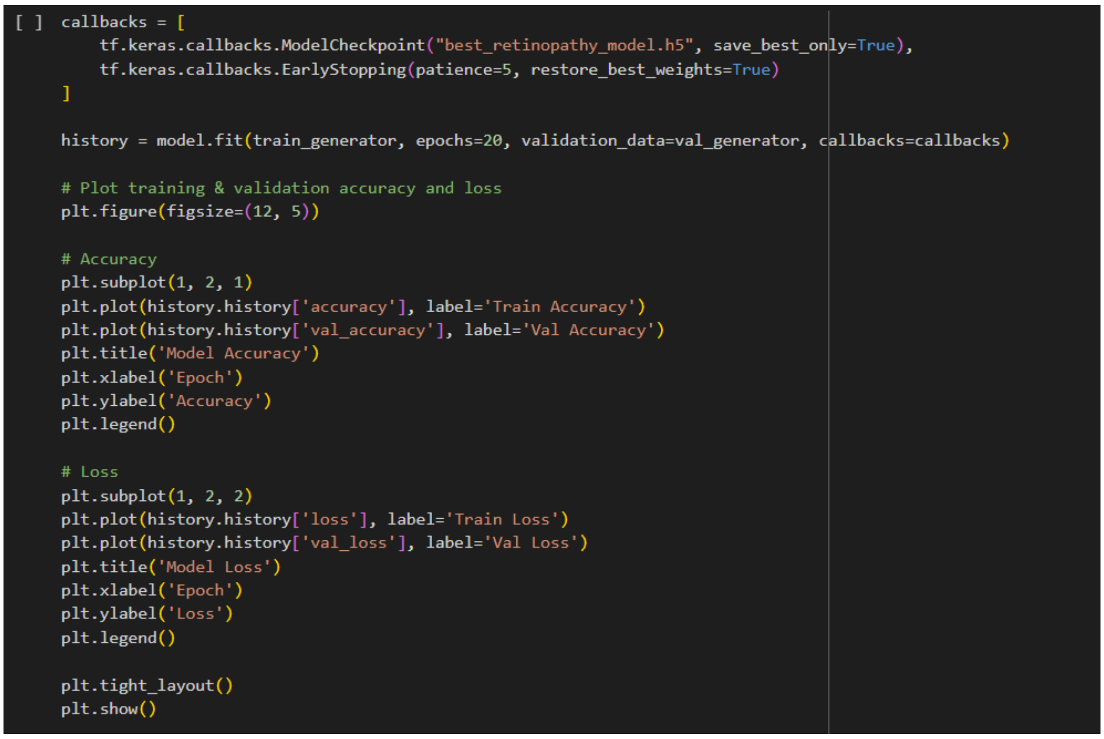
```
[ ]  splitfolders.ratio(cropped_data_path, output="/content/Final_Data", seed=42, ratio=(.8, .1, .1))
```



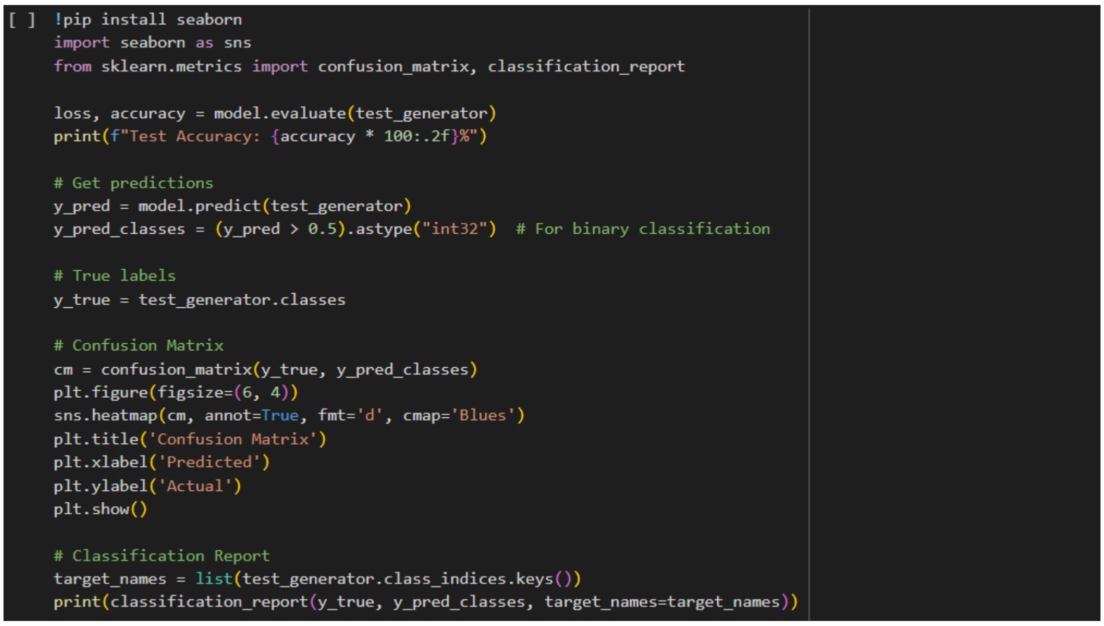
```
[ ] train_dir = '/content/Final_Data/train'
    val_dir = '/content/Final_Data/val'
    test_dir = '/content/Final_Data/test'

    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=15,
        zoom_range=0.2,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.1,
        horizontal_flip=True,
        fill_mode='nearest'
    )
    val_test_datagen = ImageDataGenerator(rescale=1./255)

    train_generator = train_datagen.flow_from_directory(train_dir, target_size=(150, 150), batch_size=32, class_mode='binary')
    val_generator = val_test_datagen.flow_from_directory(val_dir, target_size=(150, 150), batch_size=32, class_mode='binary')
    test_generator = val_test_datagen.flow_from_directory(test_dir, target_size=(150, 150), batch_size=32, class_mode='binary', shuffle=False)
```



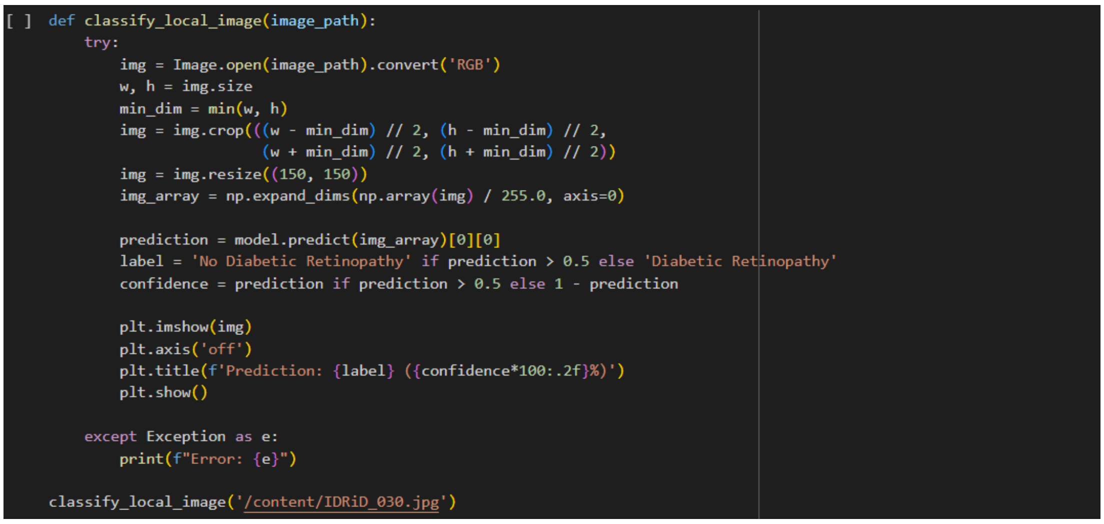
```
[ ]  model = tf.keras.models.Sequential([
         tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(128, activation='relu'),
         tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Dense(1, activation='sigmoid')
     ])

     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
     model.summary()
```



```
[ ]  callbacks = [
         tf.keras.callbacks.ModelCheckpoint("best_retinopathy_model.h5", save_best_only=True),
         tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)
     ]

     history = model.fit(train_generator, epochs=20, validation_data=val_generator, callbacks=callbacks)

     # Plot training & validation accuracy and loss
     plt.figure(figsize=(12, 5))

     # Accuracy
     plt.subplot(1, 2, 1)
     plt.plot(history.history['accuracy'], label='Train Accuracy')
     plt.plot(history.history['val_accuracy'], label='Val Accuracy')
     plt.title('Model Accuracy')
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.legend()

     # Loss
     plt.subplot(1, 2, 2)
     plt.plot(history.history['loss'], label='Train Loss')
     plt.plot(history.history['val_loss'], label='Val Loss')
     plt.title('Model Loss')
     plt.xlabel('Epoch')
     plt.ylabel('Loss')
     plt.legend()

     plt.tight_layout()
     plt.show()
```



```
[ ]  !pip install seaborn
     import seaborn as sns
     from sklearn.metrics import confusion_matrix, classification_report

     loss, accuracy = model.evaluate(test_generator)
     print(f"Test Accuracy: {accuracy * 100:.2f}%")

     # Get predictions
     y_pred = model.predict(test_generator)
     y_pred_classes = (y_pred > 0.5).astype("int32")  # For binary classification

     # True labels
     y_true = test_generator.classes

     # Confusion Matrix
     cm = confusion_matrix(y_true, y_pred_classes)
     plt.figure(figsize=(6, 4))
     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
     plt.title('Confusion Matrix')
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.show()

     # Classification Report
     target_names = list(test_generator.class_indices.keys())
     print(classification_report(y_true, y_pred_classes, target_names=target_names))
```



```
[ ]  def classify_local_image(image_path):
         try:
             img = Image.open(image_path).convert('RGB')
             w, h = img.size
             min_dim = min(w, h)
             img = img.crop(((w - min_dim) // 2, (h - min_dim) // 2,
                             (w + min_dim) // 2, (h + min_dim) // 2))
             img = img.resize((150, 150))
             img_array = np.expand_dims(np.array(img) / 255.0, axis=0)

             prediction = model.predict(img_array)[0][0]
             label = 'No Diabetic Retinopathy' if prediction > 0.5 else 'Diabetic Retinopathy'
             confidence = prediction if prediction > 0.5 else 1 - prediction

             plt.imshow(img)
             plt.axis('off')
             plt.title(f'Prediction: {label} ({confidence*100:.2f}%)')
             plt.show()

         except Exception as e:
             print(f"Error: {e}")

     classify_local_image('/content/IDRiD_030.jpg')
```

BIODATA PENULIS

My name is Reyhan Emeraldo, born in Pasuruan on 3rd April 2001. I completed my primary education at SDN 1 Petungasri, continued my junior secondary education at SMPN 1 Pandaan, and finished my senior secondary education at SMA Semesta BBS. I then pursued my higher education at the Institut Teknologi Sepuluh Nopember (ITS), where I studied in the Information Systems program. During my studies, I participated in a Double Degree Program with Universiti Teknologi PETRONAS (UTP), Malaysia, which enhanced my academic exposure in information systems and strengthened my international learning experience. Through this program, I was able to broaden my perspective on data management, business processes, and information systems implementation in a global context. In addition, I gained practical industry experience as an Operation Associate Intern at Carpedia Global Holidays Sdn. Bhd., Cyberjaya. In this role, I was involved in supporting daily operational activities, coordinating with stakeholders, managing operational data, and ensuring smooth execution of logistics-related processes. This experience strengthened my understanding of operational workflows, problem-solving in real business environments, and effective coordination within a fast-paced organization.