



**UNIVERSITY OF
KWAZULU-NATAL**

**INYUVESI
YAKWAZULU-NATALI**

Detection of Diabetic retinopathy using retinal images

By

Nhlonipho Nhlanhla Xaba

Supervisor

Prof J Tapamo

School of Engineering

**Discipline of Electrical, Electronic and
Computer Engineering - 2023**

Acknowledgment

I express my heartfelt gratitude to my supervisor, Prof Jules Raymond Tapamo, for his invaluable guidance and unwavering support during the completion of this project. His expertise and thoughtful feedback significantly contributed to the development and enhancement of this project. I would also like to extend my appreciation to the providers of publicly available APTOS and EyePACS datasets, as well as the researchers whose methodologies I adopted. Their contributions have greatly enriched the quality and depth of this work.

Declaration

I Nhlonipho Nhlanhla Xaba, hereby declare that this report titled "Detection of Diabetic retinopathy using retinal images" is the result of my own work, and it has not been submitted for any other degree or professional qualification. The sources of information and data used in this report have been duly acknowledged. I understand that plagiarism is the use of another's work without proper acknowledgment and is considered a serious academic offense. I have referenced all sources and materials used in this report, and any contributions from others have been appropriately recognized.

Nhlonipho Xaba



(Name of Supervisor)

(Student Signature)

Date: 12/11/2023

Pro. J. Tapamo

(Name of Supervisor)

(Supervisor Signature)

Date: 12/11/2023

Abstract

Diabetic retinopathy is one of the leading causes of blindness worldwide among individuals with diabetes. It is one of the major global health concerns which greatly burdens the healthcare system. It is the consequence of diabetes leading to damage of blood vessels of the retina section of the eye. If left untreated or misdiagnosed, it may lead to complete blindness. Early and accurate detection of diabetic retinopathy is therefore important to prevent patients from losing their vision. Artificial intelligence has shown remarkable success in various medical imaging tasks, and its application to diabetic retinopathy detection has gained significant attention in recent years. This report presents the conducted literature survey to explore the current state-of-the-art approaches in diabetic retinopathy detection and identifies the problem of accurately predicting the severity of diabetic retinopathy from high-resolution retina images. A convolutional neural network (CNN) based artificial intelligence approach is proposed to classify various stages of diabetic retinopathy (DR). The CNN model is designed based on the conducted literature survey, trained, evaluated, and explained in detail. The model is trained and evaluated on two datasets varying in size; the model achieved an accuracy of 95% when trained on the small dataset and 80% accuracy on the larger dataset. The model's performance is compared with current state-of-the-art models developed for DR detection. The report further explains the working principle of the system architecture and how individual components are integrated to make up the entire DR detection system. The feasibility study of the proposed solution is presented, and a work plan is developed to guide the subsequent phases of the project. The report concludes by showing the implemented web application integrated with the trained deep learning model.

Table of Contents

Abstract	iii
List of Figures	vi
List of Tables	vii
1. Introduction.....	1
2. Motivation.....	1
3. Problem Identification	1
3.1 Project description	1
3.2 Problem Formulation	1
3.2.1 Problem statement.....	1
4. Design Objectives	3
5. Literature Review.....	3
6. Constraints	5
7. Feasible Solution.....	5
7.1 Introduction.....	5
7.2 Feasible Solution Design	5
8. System Specifications	7
8.1 Functional Specifications.....	7
8.2 Non-functional Specifications.....	7
9. Feasibility Study	7
9.1 Market Feasibility	7
9.2 Technical feasibility	7
9.3 Software Feasibility	8
9.4 Hardware Feasibility	8
9.5 Feasibility Implementation	8
10. Budget.....	8
11. Workplan.....	8
12. System Architecture	9
12.1 Software Tools	10
12.1.1 Tools Used for Development	11
12.2 Data Preparation.....	12
12.2.1 Datasets Overview	12
12.2.2 Data Preprocessing.....	14
12.2.3 Data Augmentation	15
12.2.4 Data Splitting	16
12.3 Deep Learning Approach for DR Detection	16
12.3.1 Convolutional Neural Network	16

12.3.2	CNN Architecture	18
12.4	CNN Classifier - Model Design and Implementation	21
12.5	Results Analysis and Discussion.....	35
12.6	User Interface Design.....	36
12.7	Web Application Development.....	36
12.7.1	Frontend Development.....	36
12.7.2	Code Structure.....	37
12.7.3	Components	37
12.7.4	Axios	37
12.7.5	Styles	37
12.7.6	Server Implementation (Backend)	39
13.	Future Work	41
14.	Conclusion	41
	References.....	42
	Appendix.....	45

List of Figures

Figure 1: Proposed Deep Learning Framework for Diabetic Retinopathy Diagnosis	6
Figure 2: Work Plan Gantt Chart	9
Figure 3: System Architecture	9
Figure 4: Process Flow of the Proposed System	10
Figure 5: No_DR.....	13
Figure 6: Mild	13
Figure 7: Moderate.....	13
Figure 8: Severe	13
Figure 9: Proliferate DR.....	13
Figure 10: Unbalanced dataset class distribution. Class 0 is almost 49% of all images and class 3 is around 5%	13
Figure 11: No_DR (EyePACS).....	14
Figure 12: Mild (EyePACS)	14
Figure 13: Moderate (EyePACS).....	14
Figure 14: Severe (EyePACS)	14
Figure 15: Proliferate (EyePACS)	14
Figure 16: Unbalanced dataset class distribution. Class 0 is almost 73% of all images and class 4 is around 2%	14
Figure 17: The Layered Structure of Artificial Neural Network (ANN) [22]	17
Figure 18: CNN Architecture [23]	18
Figure 19: Convolutional Layer [23]	18
Figure 20: Max Pooling [21].....	19
Figure 21: Average Pooling [21]	20
Figure 22: Activation Layer [23]	20
Figure 23: Fully Connected Layer [23].....	20
Figure 24: Dense Layer [23]	21
Figure 25: Model Development Flow Chart	22
Figure 26: Architecture of Classifier	26
Figure 27: Classification Report (APTOS2019)	30
Figure 28: Confusion Matrix (APTOS2019)	30
Figure 29: Training and Validation Accuracy (APTOS2019)	31
Figure 30: Training and Validation Loss (APTOS2019)	31
Figure 31: Testing Results Visualization on APTOS2019 Dataset	32
Figure 32: Classification Report (EyePACS)	32
Figure 33: Confusion Matrix (EyePACS).....	33
Figure 34: Training and Validation Accuracy (EyePACS).....	33
Figure 35: Training and Validation Loss (EyePACS)	34
Figure 36: Testing Results Visualization on EyePACS Dataset	34
Figure 37: User Interface	36
Figure 38: Main Page of the System	38
Figure 39: Diagnostic Results Example (No DR).....	38
Figure 40: Server Initiation and Response	39
Figure 41: Backend Flow Chart	40
Figure 42: Mild DR Diagnostic Results.....	46
Figure 43: Moderate DR Diagnostic Results	46
Figure 44: Proliferate DR Diagnostic Results.....	47
Figure 45: Severe DR Diagnostic Results.....	47

List of Tables

Table 1: System Budget	8
Table 2: Development Tools.....	11
Table 3: Unbalanced dataset distribution of APTOS2019 Dataset	13
Table 4: Unbalanced Distribution of EyePACS Dataset.....	14
Table 5: Data Augmentation Used.....	16
Table 6: Proposed CNN Model.....	25
Table 7: Confusion Matrix Structure	29
Table 8: Comparative Analysis of results achieved by the proposed method with state-of-the-art using APTOS Dataset	35
Table 9: Comparative Analysis of results achieved by the proposed method with the state-of-the-art using EyePACS Dataset.....	35
Table 10: GAs Table Evidence	45

List of Acronyms

Abbreviation	Meaning
DR	Diabetic Retinopathy
CNN	Convolution Neural Network
DL	Deep Learning
UI	User Interface
GUI	Graphical User Interface
No DR	No Diabetic Retinopathy
DLCNN	Deep Learning Convolutional Neural Networks
ANN	Artificial Neural Network
GPU	Graphic Processing Unit
Web App	Web Application
Adam	Adaptive Moment Estimation

1. Introduction

Diabetic retinopathy (DR) stands as a critical complication of diabetes, affecting the blood vessels of the eye. Signs of DR start with microaneurysms, small red spots that appear when blood escapes from the retina's blood vessels. If these microaneurysms are not treated, early capillaries walls can get broken, forming haemorrhages. With the delay in treating the disease, exudates may appear on the retina, leading to permanent vision loss or vision impairment. Therefore, as the prevalence of diabetes continues to surge globally, the importance of early and accurate detection of diabetic retinopathy becomes increasingly evident. In this context, advancements in medical imaging technology, particularly the utilization of retina images, have improved the diagnosis and management of diabetic retinopathy. This report explores the field of diabetic retinopathy detection, focusing on the application of image analysis techniques to retina images. The retina, being a highly sensitive and indicative part of the eye, provides valuable information that can be utilized for early detection and intervention. By leveraging cutting-edge technologies such as deep learning and computer vision, this project aims to improve and automate the screening process for diabetic retinopathy.

2. Motivation

The motivation behind this project stems from the significant impact of diabetic retinopathy (DR) as a leading cause of vision loss worldwide. Early detection and timely intervention are critical to prevent vision impairment and improve patient care. However, the current manual diagnosis process, relying on specialized doctors, is time-consuming, subjective, and may not be readily available in all regions, leading to potential inefficiencies in diagnosing and managing DR. By developing an automated analysis system using deep learning techniques, the project aims to provide a reliable and efficient tool for DR detection, enhancing early diagnosis and enabling timely medical intervention. This project aims to contribute to improved patient outcomes, reduced risk of vision loss, and more accessible healthcare practices, ultimately positively impacting diabetic patients' lives globally.

3. Problem Identification

3.1 Project description

This project aims to develop an automated analysis system for diabetic retinopathy detection using retinal images. The project aims to harness the power of deep learning and image processing techniques to achieve this task. The primary focus is on accurately predicting the severity of diabetic retinopathy from high-resolution retina images. The system will incorporate a Graphical User Interface (GUI) to enhance practicality and user-friendliness, allowing medical professionals to upload retina images efficiently, initiate the analysis process, and receive accurate predictions of diabetic retinopathy severity. By leveraging the power of deep learning and integrating a user-friendly GUI, the project aims to enable early diagnosis, timely medical intervention, and improved patient care to prevent vision impairment and enhance the efficiency of diabetic retinopathy diagnosis in clinical settings.

3.2 Problem Formulation

3.2.1 Problem statement

Diabetic retinopathy (DR) is a leading cause of blindness among individuals with diabetes. The complexity of this problem arises from the intricate patterns and subtle features present in retinal images, which often go unnoticed by traditional diagnostic methods. Detecting and classifying these intricate patterns accurately is a daunting task. Timely diagnosis and intervention are essential for effective treatment and prevention of vision loss. Detecting DR from medical data is a cross-disciplinary domain that consists of ophthalmology and endocrinology.

Traditional Diabetic Retinopathy Detection

Traditional DR detection is done by an ophthalmologist or an optometrist through a comprehensive eye examination. The examination process includes [1]:

- Visual acuity test: Assessment of the patient's ability to see clearly at different distances.
- Pupil dilation: The eye care professional dilates the pupils of the patient through eye drops to get a clear view of the back of the eye.
- Fundus Examination: The professional then examines the eye's retina, looking for signs of diabetic retinopathy, such as changes in blood vessels, swelling, and abnormal growth of blood vessels.
- Optical coherence tomography (OCT): Non-invasive imaging using light waves to capture cross-sectional retina images, aiding in the detection of diabetic retinopathy and other eye conditions.
- Diagnosis and Treatment: Based on the examination results, the eye care professional proceeds to diagnose diabetic retinopathy and recommend treatment plan, which normally involves monitoring, laser treatment, medication, or surgery.

This diagnostic method is manual, time-consuming, and sometimes inefficient. To overcome these challenges, this project leverages the power of deep learning and sophisticated image processing techniques to create an efficient and automated system. The methods employed are designed to automatically extract and analyze features within retinal images, enabling precise classification into multiple categories representing different stages of diabetic retinopathy and healthy retina.

The key challenges to be tackled in this project include the following:

1. **Dataset Collection and Preprocessing:** Acquiring a comprehensive dataset of retinal images with diverse representations of diabetic retinopathy stages and healthy retinas. Ensuring proper preprocessing to enhance image quality, remove artifacts, and standardize the dataset.
2. **Feature Extraction:** Designing a model that is capable of automatically extracting meaningful and relevant features from retinal images to differentiate between the various stages of diabetic retinopathy accurately.
3. **Class Imbalance:** Addressing the issue of class imbalance in the dataset, as certain stages of diabetic retinopathy may be underrepresented, potentially leading to biased model predictions.
4. **Performance Metrics:** Establishing appropriate performance evaluation metrics to assess the accuracy, sensitivity, specificity, and AUC-ROC of the deep learning model in diagnosing diabetic retinopathy.
5. **User Interface and Integration:** Creating a user-friendly interface allows healthcare professionals to interact with the trained model conveniently, enabling automated and timely DR diagnosis.

Based on the success of deep learning evident in the literature review, the ultimate goal is to investigate its application to diabetic retinopathy detection. By overcoming the challenges mentioned above, the proposed solution will contribute to early detection and effective management of diabetic retinopathy, potentially preventing vision loss and improving patient care in the field of ophthalmology.

4. Design Objectives

The main objectives of this project are to:

- **Develop an automated analysis system:** Create an automated system that analyzes high-resolution retina images to detect diabetic retinopathy. The system will reduce manual intervention and enhance diagnostic efficiency.
- **Utilize deep learning techniques:** Based on deep learning prevalence, the aim is to leverage its ability to learn complex patterns and features from retina images, enabling accurate diabetic retinopathy severity predictions.
- **Design a Graphical User Interface (GUI):** Integrate a user-friendly GUI into the system to provide medical professionals with an intuitive platform for interacting with the diagnostic tool. The GUI will facilitate easy image uploading, analysis initiation, and result retrieval.
- **Enhance early diagnosis:** Enable early detection of diabetic retinopathy through the automated analysis system, allowing healthcare providers to intervene promptly and prevent vision loss in diabetic patients.
- **Improve patient care:** By providing quick and reliable diabetic retinopathy severity predictions, the system will aid medical professionals in making informed decisions and tailoring personalized patient treatment plans.
- **Enhance healthcare practices:** Contribute to advancing diabetic retinopathy diagnosis by introducing an efficient and accurate diagnostic tool that can be integrated into clinical workflows, ultimately improving patient outcomes.

5. Literature Review

Authors are continuously working on the improvement diagnosis of diabetic retinopathy using cutting-edge technologies. This section reviews the most recent works on diabetic retinopathy detection using various artificial intelligence techniques.

Yadav et al. [2] proposed a computer vision-based technique to analyze and predict diabetic retinopathy based on retina images. In their work, the CNN-based model is investigated. The initial step of the model involves the preprocessing of the image, conversion to greyscale, and application of a canny edge detection algorithm to perform optical disk segmentation. The final stage involves vessel extraction, and the results are fed to the CNN model for training. Their developed framework achieved an accuracy of 98.50%, outperforming the Support Vector Machine (SVM) with an accuracy of 87.40%, as evaluated in comparative testing.

In [3], a method of detection of diabetic retinopathy is proposed by Zaaboub and Douik who detect and segment the exudate using a random forest classifier. This proposed method takes the original retina image, removes the optical disk, performs feature selection, and results are fed to a random forest classifier. Their method achieved remarkable results, including a specificity of 99.5%, an accuracy of 99.50%, and an area under the AUC curve of 0.952.

Harikrishna and Kumar [4] implemented a convolutional neural network for classifying multiple stages of diabetic retinopathy. Firstly, hybrid features are derived from the IDRID dataset using Local Binary Pattern (LBP), Local Gaussian Difference Extrema Pattern (LGDEP), and Histogram of Oriented Gradient (HOG) descriptors. Subsequently, Linear Discriminant Analysis (LDA) is applied to select the most discriminative features based on their inter and intra-disease dependencies. Finally, the obtained features are utilized to train the DLCNN (Deep Learning Convolutional Neural Network) model, enabling the classification of diabetic retinopathy grades for each test retinal image. Their work

achieved an accuracy of 96.97%, precision of 98.98%, sensitivity of 97.30%, specificity of 98.98 and F1-score of 97.37%.

A computer-aided diagnosis based on digital image processing was proposed by Enrique et al. [5] for early detection of diabetic retinopathy from retina images. The primary objective was to automate the classification of non-proliferative diabetic retinopathy grades in retinal images. The steps involved image processing to identify and isolate blood vessels, microaneurysms, and hard exudates. Subsequently, features are extracted and fed into a support vector machine to determine the retinopathy grade for each retinal image. Their proposed work demonstrated a predictive capacity of 94% and a maximum sensitivity of 95%, highlighting its potential for early diabetic retinopathy detection.

Misgina et al. [6] developed a smartphone-based application to diagnose diabetic retinopathy. The author employed the convolutional neural network and binary decision tree to detect and classify diabetic retinopathy from retina images. The developed model was integrated into a smartphone application to facilitate the classification of retina images. Thereby providing convenience and ease of use for healthcare professionals and patients alike. The outcome was remarkable, with the proposed model achieving exceptional performance metrics, including an accuracy of 99.86%, sensitivity of 99.25%, and specificity of 99.60%.

A deep learning model to perform early stages detection of diabetic retinopathy and facilitate the screening process was created by Nasir et al. [7]. The proposed model was based on convolutional neural networks and can classify diabetic retinopathy between two classes, whether diabetic retinopathy is present or not, based on the retina images dataset. The model was evaluated using various performance metrics. The Convolutional Neural Network proposed achieved a training accuracy of 94%, testing accuracy of 96.0%, and sensitivity of 82.9%. However, the authors acknowledged a limitation regarding the model's performance in real-world scenarios with non-identical photos and variations in environmental and hardware issues, emphasizing the need for robustness and adaptability in such applications.

In their study, Nagaraj et al. [8] proposed a deep learning framework for diabetic retinopathy diagnosis. The research employs a customized version of a commonly used Convolutional Neural Network (CNN) to address fundus image classification challenges related to diabetic retinopathy. The suggested framework provides a rapid and effective assessment of diabetic retinopathy presence and severity. The model was trained using 35126 images using the proposed VGG16 CNN architecture. It swiftly determines whether an individual has DR and promptly reports the disease severity level if detected. The model was tested on 2938 images, and the highest test accuracy obtained was 73.72%.

A deep learning model to assist ophthalmologists in clinical diagnosis and enabling detection and classification of diabetic retinopathy was developed by Vipparthi et al. [9]. The proposed model comprises three phases: image enhancement, feature extraction, and detection and classification. Feature extraction involves the extraction of exudates and blood vessels. The Res-Block model is applied to classify and diagnose diabetic retinopathy from the retina fundus images. In their evaluation, the model achieved an impressive accuracy rating of 92.33%.

Kumar and Babu [10] investigated three deep learning models, namely Densenet-169, ConvLSTM, and Dense-LSTM, to detect diabetic retinopathy. These models were compared for their efficiency in performing early detection of diabetic retinopathy and determining its severity. The results indicated that the ConvLSTM model outperformed the others, achieving an impressive accuracy rate of 99%. Densenet-169 followed closely with an accuracy of 94%, while the hybrid model Dense-LSTM achieved an accuracy of 83%.

Sabir and Umma [11] conducted the survey to evaluate the effectiveness of deep learning approaches for diabetic retinopathy detection. The authors performed a comprehensive systematic review, including meta-analysis encompassing various deep learning methods and classifiers from recent research. The

authors comparative analysis is based on the accuracy, precision, sensitivity, recall and f1-score performance measures. The deep learning models the study analyses are trained on different datasets. The analysis results show that the convolutional neural network (CNN) model proposed by Nasir et al. [7] demonstrated exceptional performance amongst other models.

6. Constraints

- Time: Limited time to design and implement the system.
- Budget: Allocated standard budget to cater for expenses concerning the project.
- Data Availability: Limited data may affect the model's performance.
- Data Quality: Retina images may contain artefacts, variations in lighting conditions, or other imperfections that could impact the model's accuracy.
- Healthcare Expertise: Limited access to experienced healthcare practitioners (Optometrists and Ophthalmologist) could pose challenges in model development and implementation.
- Hardware Constraints: Limited availability of powerful GPUs or cloud computing services may have an impact on training of the model.
- Real-World Validation: Due to differences in imaging technology, lighting conditions, and patient demographics, assessing the model's performance in a real-world clinical setting may be difficult.

7. Feasible Solution

7.1 Introduction

Current best practices in medical image analysis, as identified in the literature survey, indicate that a feasible solution for developing an automated diabetic retinopathy detection system is to harness the power of deep learning. The system will incorporate a user-friendly Graphical User Interface (GUI) to streamline image uploading, analysis, and result retrieval for healthcare professionals.

7.2 Feasible Solution Design

Based on the literature review conducted, it is evident that leveraging deep learning techniques, particularly convolutional neural networks, shows great promise in the field of diabetic retinopathy detection. Figure 1 below illustrates the system architecture, which draws inspiration from the solutions presented in the literature sources. In [6], [4], and [7], several key architectural elements were identified to form the foundation of the proposed model. Nasir et al.'s work [7] serves as foundational inspiration. It contributed to the design by introducing a model based on convolutional neural networks for early DR detection, achieving the highest accuracy, as evident in the survey conducted by Sabir and Umma [11]. Dharsinala and Udaya's work [4] provided valuable insights regarding the preprocessing steps, augmentation and feature extraction techniques. The development of a user interface integrated with deep learning for effective patient diagnosis draws inspiration from the work of Misgina et al. [6]. Their innovative approach guides the creation of a transformative tool for healthcare professionals. This architecture serves as the foundation for developing the diabetic retinopathy detection system.

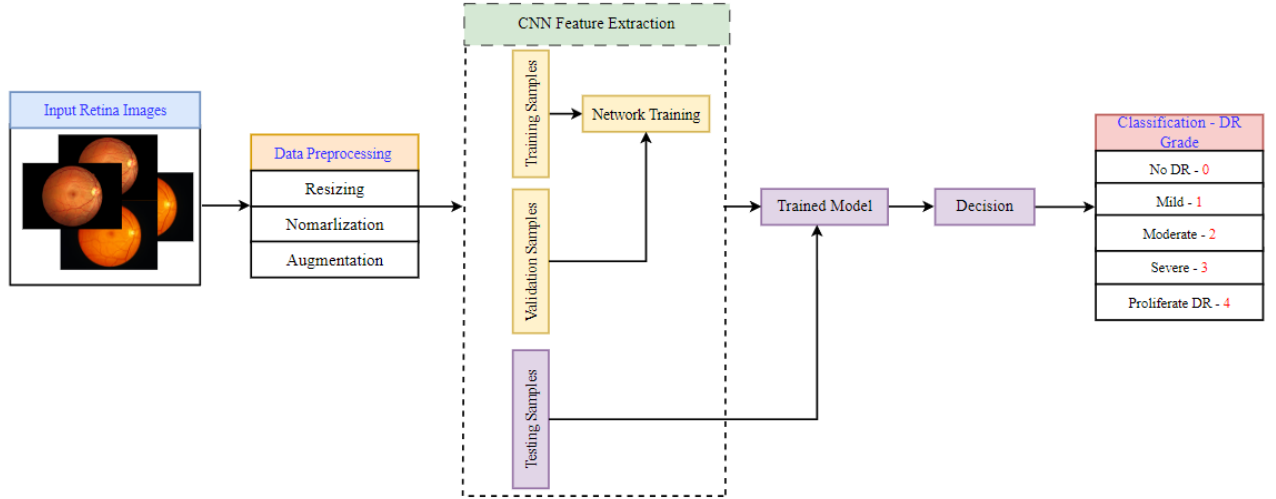


Figure 1: Proposed Deep Learning Framework for Diabetic Retinopathy Diagnosis

An input retina image is uploaded via the user interface provided. The image preprocessing step involves processing the image to standard size and normalization. The generation of augmented images is done to enhance the training process. The augmented images are used to expand the dataset for training the CNN model and help to improve model generalization. As the CNN trains using the training set, it automatically learns to extract hierarchical features from retinal images. The extracted validation features are used to validate the classification algorithm. The trained model is, therefore, used to classify DR grade.

The decision to employ Convolutional Neural Networks (CNNs) for diabetic retinopathy detection is substantiated by a comprehensive literature survey, which underscores the superior performance of CNN-based methods in medical image analysis, particularly in the context of retinal images. Numerous studies have reported consistently higher accuracy levels achieved by CNNs in detecting diabetic retinopathy severity compared to alternative techniques.

The ability of CNNs to automatically learn and represent intricate spatial features from raw pixel data plays a pivotal role in their success. This power of CNN is particularly evident in medical image analysis, where nuances in textures, structures, and patterns provide valuable diagnostic information. CNN-based methods demonstrated the capability to capture these complex features and, in turn, significantly improved the accuracy of diabetic retinopathy severity predictions. Notably, this negates the need for labour-intensive manual feature engineering, a characteristic that distinguishes CNNs from traditional methods.

While alternative approaches such as Support Vector Machines (SVMs), Random Forests, and conventional image processing techniques remain valuable in various contexts, their limitations in capturing complex spatial information within retinal images led to comparatively lower accuracy levels. Moreover, the adaptability of CNN architectures to large datasets ensures robustness and scalability, aligning seamlessly with the diverse and extensive data samples often encountered in medical image analysis.

8. System Specifications

8.1 Functional Specifications

- **Image Data Ingestion:** The system should be capable of ingesting retinal images in various formats, including JPEG and PNG.
- **Image Selection:** The system must allow the healthcare professionals to select retina image of their choice to analyse.
- **Pre-processing:** The system must include pre-processing steps namely, resizing images to 224x224, and normalization, to prepare images for analysis.
- **Feature Extraction:** The system should be capable of extracting relevant features such as haemorrhages, exudates, microaneurysms, and blood vessel patterns using deep learning techniques.
- **Classification:** System must be able to classify retina images into different stages of diabetic retinopathy (No DR, mild, moderate, severe, proliferative DR) using deep learning techniques.
- **Confidence Score:** System should provide confidence score with each retinal image analysis to indicate the level of certainty in the classification results. This score should be displayed to healthcare professionals for assessment and decision-making.
- **Automated Analysis Initialisation:** The system should automatically initiate the analysis process upon successful upload of the retina image.
- **Model Training and Validation:** The system should allow for training and validation of the deep learning model using labelled data.

8.2 Non-functional Specifications

- **High Accuracy:** The system should have a minimum accuracy of 95% in detecting the presence of diabetic retinopathy.
- **Reliability:** The system should be highly reliable, minimizing downtime and errors that could affect patient care.
- **Speed and Efficiency:** The system should perform image analysis and classification within 60 seconds of image upload to minimize waiting time for healthcare professionals.
- **User-Friendly Interface:** The user interface should be intuitive and user-friendly by providing a seamless and easily navigable experience, ensuring that medical practitioners can efficiently interact with the system to access and analyse retinal images.

9. Feasibility Study

9.1 Market Feasibility

The market feasibility assessment of the automated diabetic retinopathy detection system reveals a significant demand within the healthcare sector. Diabetic retinopathy is a prevalent condition affecting a large population, necessitating efficient and accurate diagnostic tools. The system's ability to provide timely severity predictions aligns well with the needs of ophthalmologists and retinal specialists. The increasing adoption of artificial intelligence and deep learning in healthcare diagnostics indicates a favourable market trend. The system addresses a critical medical need, which enhances its potential for wide acceptance and utilization within the healthcare community.

9.2 Technical feasibility

From a technical perspective, the project is deemed feasible. Convolutional Neural Networks (CNNs) have demonstrated remarkable success in image analysis tasks, including diabetic retinopathy detection. The availability of open-source deep learning frameworks like TensorFlow and PyTorch provides a strong foundation for model development. Adequate computational resources, such as Google free GPUs and free development software such as Anaconda, ensure the

feasibility of training and deploying the CNN model. Additionally, integrating a user-friendly Graphical User Interface (GUI) for healthcare professionals is well-supported by modern web and mobile app development frameworks.

9.3 Software Feasibility

The software feasibility analysis indicates a high level of viability. The project leverages established deep learning libraries, enabling the implementation of the CNN model. Preprocessing techniques for image enhancement and normalization are readily available. Developing a GUI tailored to the healthcare environment can be achieved using popular cross-platform frameworks like React Native or Flutter. These frameworks provide efficient tools for building responsive and intuitive interfaces. Overall, the software components required for the project are well within reach and supported by a robust developer ecosystem.

9.4 Hardware Feasibility

The hardware feasibility assessment confirms the project's viability. While deep learning models can be resource-intensive, the availability of GPUs and cloud computing services offers the computational power required for training and inference. Modern smartphones and computers possess adequate processing capabilities to run the GUI app seamlessly. The project does not demand specialized hardware, making it accessible for healthcare professionals using standard computing devices.

9.5 Feasibility Implementation

The feasibility study culminates in a positive feasibility implementation outlook. With favourable market conditions, adequate technical resources, and suitable software and hardware components, the project demonstrates a high likelihood of successful implementation.

10. Budget

This section outlines the allocation of financial resources across various categories, including hardware and software. It serves as a critical component in ensuring the feasibility and cost-effectiveness of the project. The budget Table 1 below provides a clear breakdown of estimated costs.

Table 1: System Budget

Description	Quantity	Price
Main Softwares - Jupyter Notebook and VSCode	1	R0.00

As shown in Table 1, the project maintains a cost-effective approach. The necessary software tools for development are free, although premium versions of these development environments are accessible but the free versions should suffice for the development of the system.

11. Workplan

Figure 2 provides the Gantt chart that outlines the project planning and timeline for the development of the automated diabetic retinopathy detection system. This visual representation offers an overview of the various project phases, tasks, and their scheduled durations.

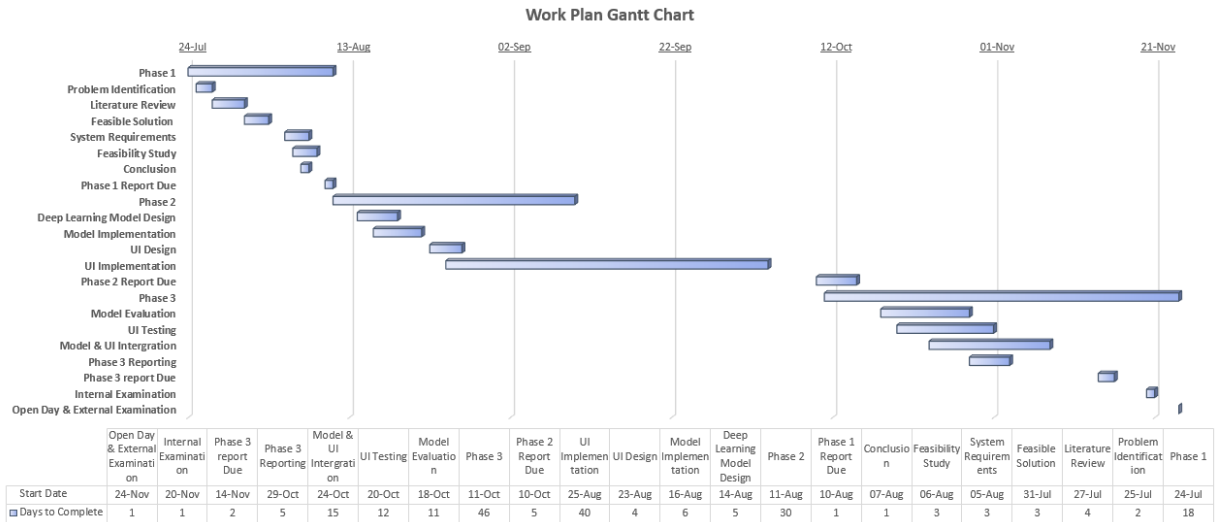


Figure 2: Work Plan Gantt Chart

12. System Architecture

Figure 3 below shows the overall system architecture. The user (healthcare practitioner) uploads the image through the web app. It is designed to facilitate the efficient and user-friendly detection of diabetic retinopathy from retinal images. This architecture consists of three primary components: The Web Application, the Fast API, and the model block.

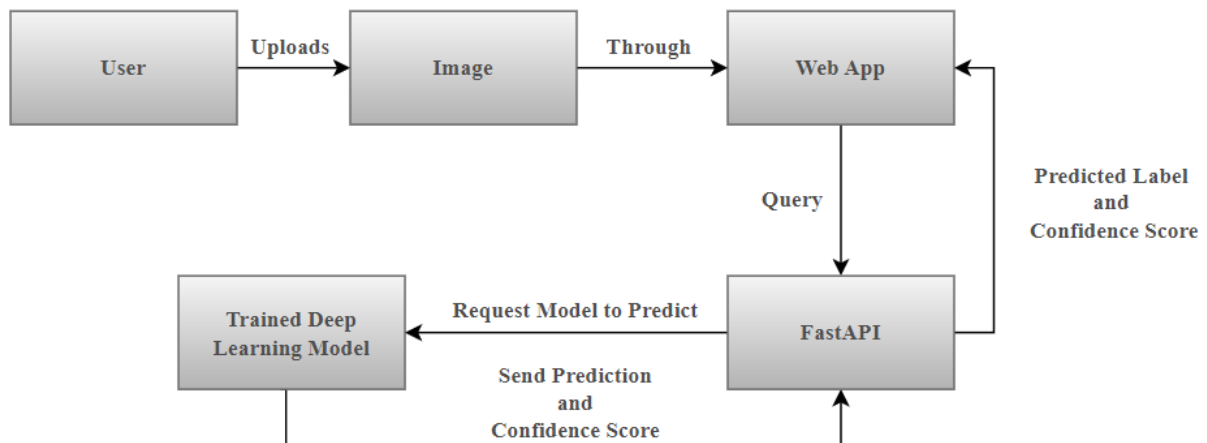


Figure 3: System Architecture

- **Web Application:** At the forefront of the system is the web application. This is the user interface which provides healthcare professionals with an intuitive platform for easy interaction with the system. The user uploads the retina images through the provided web app user interface, and the system automatically initiates the prediction process.
- **Fast API:** This part of the system serves intermediary between the web app and the deep learning model. FastAPI is a Python web framework. Once an image is uploaded through the Web app, it is sent to the FastAPI as a query, where it is pre-processed and prepared for analysis. The FastAPI manages the data flow between the user interface and the deep learning model.
- **Trained DL Model:** This is the system's main component that encompasses the Convolutional Neural Network (CNN) designed for diabetic retinopathy detection. This component processes the retinal images using advanced deep learning techniques. The image, pre-processed by the

Fast API, undergoes analysis within the CNN model. The model conducts a comprehensive assessment of the retinal image, generating a prediction and confidence score related to the presence and severity of diabetic retinopathy.

Data flow

1. **Image Upload:** The process begins when a healthcare professional uploads a retinal image through the Web Application (Web app). The image is sent to the FastAPI, where it is ready for analysis.
2. **Preprocessing:** The FastAPI performs initial preprocessing steps, preparing the image for analysis using the deep learning model.
3. **Model Analysis:** The prepared image is then passed to the trained DL model, where the CNN-based model thoroughly examines the image and generates predictions.
4. **Prediction and Confidence:** Upon analysis, the model returns the prediction regarding the presence and severity of diabetic retinopathy. Additionally, it provides a confidence score, indicating the level of certainty in the prediction.
5. **Result Display:** The final prediction and confidence level are returned to the FastAPI, which, in turn, presents this information through the Web Application. Healthcare professionals can view the diagnostic outcome and the associated confidence score, ensuring transparency and facilitating clinical decision-making.

Figure 4 presents the process flow of the proposed system.

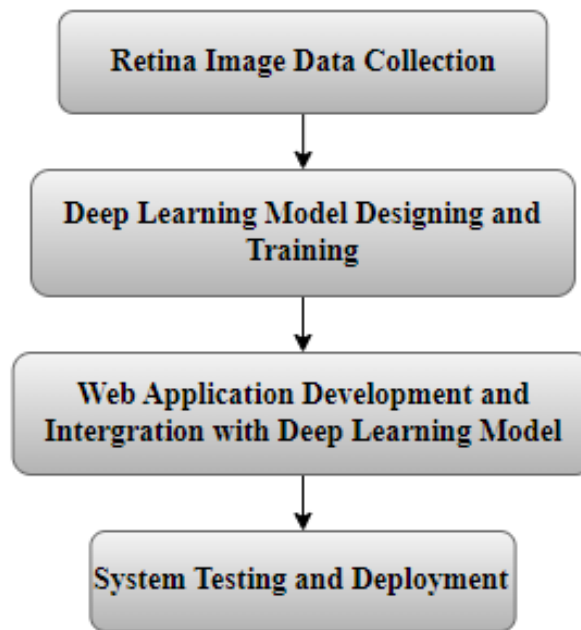


Figure 4: Process Flow of the Proposed System

12.1 Software Tools

This section provides an overview of the software, development tools, and frameworks employed during the project's development phase. It encompasses the critical tools instrumental in building and implementing the system for diabetic retinopathy detection. The core software tools used for development are summarized in Table 2 below.

12.1.1 Tools Used for Development

Table 2: Development Tools

Tool	Description/Purpose
Integrated Development Environments (IDEs)	
Jupyter Notebook	Jupyter Notebook is the integrated development environment (IDE) for writing and executing code. It allowed for interactive coding, data visualization, and easy project progress documentation.
Visual Studio Code	Visual Studio Code (VSCode) is a lightweight yet powerful source-code editor developed by Microsoft. This IDE is used for writing fronted code of the system.
PyCharm	PyCharm is a renowned Integrated Development Environment (IDE) designed specifically for Python development. Developed by JetBrains, PyCharm provides a comprehensive and user-friendly environment. This IDE is used to write the backend code of the system
Programming Languages	
Python	Python is chosen as the primary programming language for its versatility and an extensive array of libraries and frameworks suitable for deep learning and image processing tasks. Python serves as the foundation for coding and data manipulation throughout the project
React	React, developed and maintained by Facebook, is a powerful and widely adopted JavaScript library for building user interfaces, particularly for single-page applications. In this project, this programming language is used to develop the user interface (website) of the system.
Frameworks and Libraries	
TensorFlow	This is a Google-developed open-source deep learning framework. This framework serves as the foundation for the deep learning model. The deep learning model is designed, trained, and evaluated using the power of this library.
Keras	Keras is a component of the TensorFlow framework. It is used in this project because of its high-level neural network API, which makes building complex deep learning architectures easier. Keras offers an easy-to-use interface for creating the CNN model.
matplotlib	This is a data visualization library static, interactive, and animated plots in Python. It is used in this project to visualize images and predicted labels through plots.
seaborn	This is a data visualization library built on top of Matplotlib library. It is used to designed to create informative and attractive statistical graphics in Python. It is used in this project to visualize

	during evaluation of project to visualize confusion matrix.
sklearn	This is the scikit-learn library commonly used for development of machine learning models. It is used for various purposes in this project including building of CNN model.
OpenCV	Open Source Computer Vision Library, often referred to as cv2 in Python, is a widely used open-source computer vision and image processing library.
numpy	Numerical Python, is a fundamental library for numerical computing in Python. This library provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.
pandas	This is an open-source data analysis and manipulation python library.
Code Version Control	
Github	GitHub is a web-based platform for version control and collaboration, primarily used for hosting and managing software development projects. In this project is used to maintain versions of deep learning models developed, frontend and backend code of the system.
Other	
Postman Software	This software is a widely used tool for API development, testing, and documentation. In this project the software is used to test the GET API requests from the website to local server.
Figma	This is a web-based tool used for designing and prototyping. In this project this tool was used to design the user interface (frontend) of the system.

12.2 Data Preparation

Data preparation is crucial when developing a Convolutional Neural Network (CNN) model or any machine learning model. It involves collecting, preprocessing, and organizing the data to make it suitable for training and testing the CNN model [12].

12.2.1 Datasets Overview

Datasets are the essential raw material for training, testing, and evaluating deep learning models, particularly in Convolutional Neural Networks (CNNs). The training dataset trains CNNs to recognize and understand patterns, features, and objects within images, enabling the models to perform tasks such as image classification.

12.2.1.1 APTOS2019 Dataset

The first dataset used in this project is Blindness Detection APTOS-2019 dataset. This dataset publicly available on the Kaggle website [13]. All images are saved into their respective folders according to the severity/stage of diabetic retinopathy using the train.csv file provided. In the train.csv file provided, each image is labelled as {0, 1, 2, 3, 4}, depending on the disease's severity. All images are in .png format.

There are five directories with the respective images:

0 - No_DR: Images indicating the absence of diabetic retinopathy.

1 – Mild: Images showing mild diabetic retinopathy.

2 – Moderate: Images with moderate diabetic retinopathy.

3 – Severe: Images indicating severe diabetic retinopathy.

4 - Proliferate_DR: Images representing proliferative diabetic retinopathy.

Some samples from different classes of APTOS2019 are shown in Figure 5-Figure 9 below.



Figure 5: No_DR



Figure 6: Mild



Figure 7:
Moderate

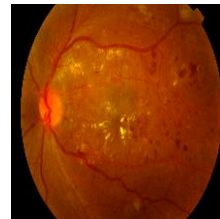


Figure 8: Severe

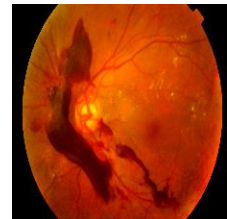


Figure 9:
Proliferate DR

Table 3: Unbalanced dataset distribution of APTOS2019 Dataset

Class	Name	Number of Images	Percentage
0	No_DR	1805	49.29%
1	Mild	370	10.10%
2	Moderate	999	27.28%
3	Severe	193	5.27%
4	Proliferate_DR	295	8.06%

Figure 10 below illustrates the number of images available for each class.

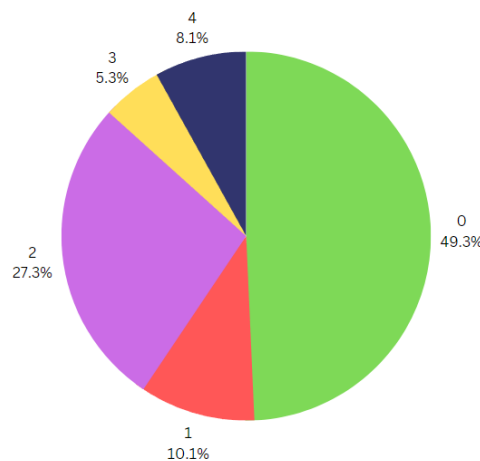


Figure 10: Unbalanced dataset class distribution. Class 0 is almost 49% of all images and class 3 is around 5%

Figure 10 above shows the distribution of retina images of the APTOS2019 dataset. As seen in the figure, approximately half of the dataset is from class 0 (No DR) with the class 3 having the least number of images.

12.2.1.2 EyePACS Dataset

The second dataset used in this project is also publicly available on Kaggle. This dataset consists of 35126 retina images and it is provided by EyePACS. The dataset is structured as APTOS2019 dataset. All images are in .jpeg format.

Some samples from different classes of EyePACS Dataset are shown in Figure 11-Figure 15 below.

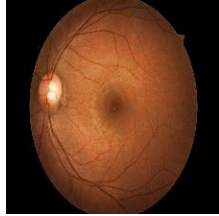


Figure 11:
No_DR
(EyePACS)



Figure 12: Mild
(EyePACS)



Figure 13:
Moderate
(EyePACS)

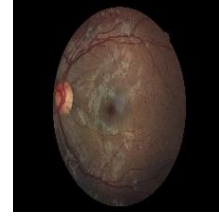


Figure 14: Severe
(EyePACS)

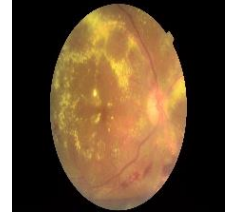


Figure 15:
Proliferate
(EyePACS)

Table 4: Unbalanced Distribution of EyePACS Dataset

Class	Name	Number of Images	Percentage
0	No_DR	25810	73.48%
1	Mild	2443	6.95%
2	Moderate	5292	15.07%
3	Severe	873	2.49%
4	Proliferate_DR	708	2.02%

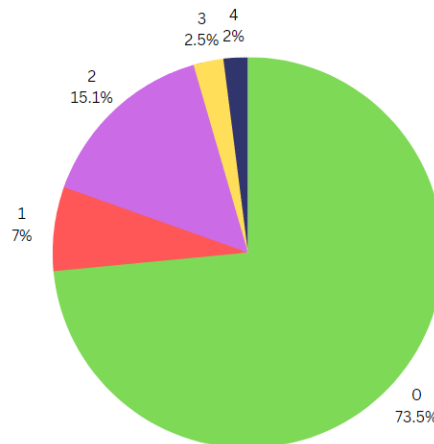


Figure 16: Unbalanced dataset class distribution. Class 0 is almost 73% of all images and class 4 is around 2%

Figure 16 above shows the distribution of retina images of the EyePACS dataset. More than half of the dataset is from class 0 (No DR) with the class 4 having the least number of images. This dataset shows that there's far greater data imbalance in it compared to the APTOS2019 dataset.

12.2.2 Data Preprocessing

Data preprocessing describes the set of modifications and procedures performed on input data before feeding it into a neural network for inference or training. Making the data compatible with the network architecture, increasing training efficiency, and enhancing the model's capacity to learn and generalize from the data are the main objectives of the data preprocessing step in computer vision [14].

12.2.2.1 Image Resizing

Images in a dataset both datasets come in various sizes. Resizing them to a consistent resolution is important for the neural network to process them. All the images in the datasets are scaled to a fixed width and height of 224 pixels. Images with smaller sizes require less computational resources for processing.

12.2.2.2 Image Normalization

Normalization of image pixel values involves transforming an image's pixel values to a specific range, such as 0-1 or -1 to 1. This can be done using a variety of methods, described in [15] and [16]:

- **Min-max normalization:** This technique involves subtracting the minimum pixel value from each pixel and then dividing by the range, which is the difference between the maximum and minimum pixel values. The pixel values are then normalized to range between 0 and 1.
- **Z-Score normalization:** This method divides each pixel value by the standard deviation of the pixel values and subtracts the mean pixel value from each pixel value.
- **Per-Channel Normalization:** In color images with multiple channels (e.g., RGB), it's common to apply per-channel normalization. Each color channel is normalized separately using one of the above techniques.

The normalization technique adopted is Min-max normalization, as it is simple to implement and is effective for normalizing data with a wide range of values, such as retina images. Each pixel in an image has an intensity value, and normalizing these values helps ensure that the neural network converges faster during training.

The Min-Max normalization is given by the following equation:

$$x_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

Where;

X – Is the original pixel value

X_{min} – Is the minimum pixel value

X_{max} – Is the maximum pixel value

$x_{normalized}$ – Is the normalized pixel value

12.2.3 Data Augmentation

Data augmentation is a technique for artificially increasing the size of a training dataset by producing modified versions of existing images. This can be accomplished by using several techniques such as cropping, flipping, rotating, and adding noise. It is important to deep learning because it prevents the model from overfitting the training data. Overfitting occurs when the model learns the individual features of the training data so well that it cannot generalize to the data it has never seen. By augmenting the training data, the model is forced to learn more broad traits that are more likely to be present in the newly received data [17]. Data augmentation is also one of the methods used to handle data imbalance. As shown in Figure 16 and Figure 10, both datasets have an imbalanced number of images across five classes; therefore, data augmentation is employed to balance these datasets.

12.2.3.1 Handling Dataset Imbalance of APTOS2019 Dataset

In the APTOS2019 dataset, the class with highest number of images is class 0, with 1805 images. Therefore, data augmentation must be applied in class 1 – 4. Table 5 below show the data augmentation techniques adopted from [18] and [19] and applied to this dataset to balance this dataset. The Keras Processing Layer's *ImageDataGenerator* was used to facilitate the augmentation process.

Table 5: Data Augmentation Used

Argument	Parameter Value
Zoom Range	0.2
Width Shift Range	0.2
Rotation Range	50
Horizontal Flip	True
Vertical Flip	True
Fill Mode	Nearest

As the result of the data augmentation, the dataset was expanded to 9025 images, each class consisting of 1805 images.

12.2.3.2 Handling Dataset Imbalance of EyePACS Dataset

The EyePACS Dataset has a huge imbalance between class 0 and other classes. Performing data augmentation until every class has 25810 images as in class 0, is computationally expensive and might lead to overfitting of the model since other classes like class 4 have far fewer images (708). This raises the need to downsample class 0 and perform the same data augmentation techniques applied in the APTOS2019 dataset to the remaining classes. Class 0 is downsampled such that the number of images is equal to 10000; this number is chosen in consideration of the available computational resources. This down sampling and augmentation process resulted in a dataset of 50000 images, ensuring a balanced representation of 10000 images per class.

12.2.4 Data Splitting

Dataset splitting divides a dataset into subsets for training, validating, and testing a deep learning model. The dataset in this project is split as follows [20]:

- **Training set:** The training set trains the deep learning model. The model learns to predict the output variable for the training data by adjusting its parameters. 80% of the dataset is used for training.
- **Validation set:** The validation set is used to evaluate the performance of the trained model on unseen data. 10% of the dataset is used for validation purposes.
- **Test set:** The test set is used to evaluate the final performance of the trained model on unseen data. The model's performance on the test set is a good estimate of how the model will perform on real-world data. 10% of the dataset is used for testing the model.

12.3 Deep Learning Approach for DR Detection

In the field of diabetic retinopathy detection, deep learning serves as a powerful tool, particularly through Convolutional Neural Networks (CNNs); this is evident in the literature survey section of this report. This section delves into the foundational working principles, architecture complexities, and the diverse layers of CNNs. The theoretical exploration lays the groundwork for a solid understanding of the CNN model designed for classifying retina images based on DR severity.

12.3.1 Convolutional Neural Network

Convolutional Neural Networks are a type of deep neural network developed to analyse grid-like input like images and videos. According to the literature review, the CNN architecture has proven to be highly effective in computer vision tasks, making it a good choice for analysis of retinal images.

An artificial neural network (ANN) is a machine-learning model inspired by the structure and function of the human brain. ANNs are composed of interconnected nodes known as neurons that process information and communicate with one another.

In a regular Neural Network, there are three types of layers [21]:

- **Input Layer:** This is where the input to the model is provided. The number of neurons in this layer is equivalent to the total number of features in the data, e.g., the number of pixels in an image.
- **Hidden Layer:** The output of the input layer is sent to the hidden layer. The number of hidden layers varies depending on the model and the data type. Each hidden layer may comprise a different number of neurons, which is usually greater than the number of features. The output of each layer is determined through matrix multiplication of the previous layer's output with the layer's adjustable weights, followed by the addition of adaptable biases. Afterward, an activation function is used to introduce non-linearity into the network.
- **Output Layer:** The output of the hidden layer is transmitted to a logistic function such as sigmoid or softmax, which converts the output for each class into a probability score for that class.

Figure 17 below shows the typical structure of the artificial neural network.

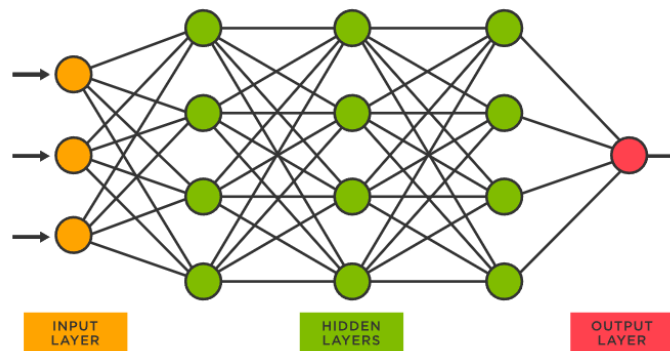


Figure 17: The Layered Structure of Artificial Neural Network (ANN) [22]

A Convolutional Neural Network (CNN) is an extended form of artificial neural networks (ANNs) primarily designed for feature extraction from grid-like matrix data. It is particularly suited for visual datasets such as images and videos with prevalent data patterns. A CNN transforms an input image into a feature map, which is then processed through multiple convolutional and pooling layers to produce a predicted output. This distinguishes it from traditional feature extraction methods.

12.3.1.1 CNN Working Principle

A convolutional neural network begins by taking an input image, which is subsequently converted into a feature map by passing through a sequence of convolutional and pooling layers. In the convolutional layer, a group of filters is applied to the input image, with each filter generating a feature map that represents a particular aspect of the input image. Following this, the pooling layer reduces the size of the feature map while preserving the essential information [23]. Figure 18 shows a typical CNN architecture.

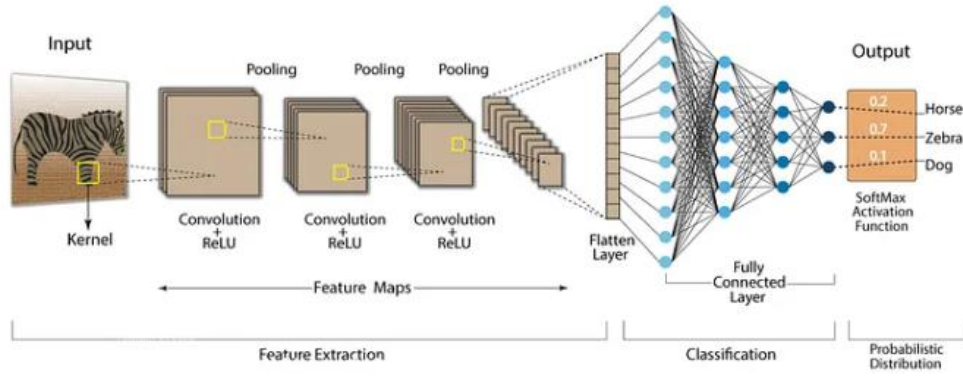


Figure 18: CNN Architecture [23]

12.3.2 CNN Architecture

The input, hidden, and output layers are the three main components of a conventional CNN architecture. The image is received by the input layer, which then sends it to the hidden layers, which are made up of numerous convolutional and pooling layers. The output layer provides a predicted class label or probability scores for each class.

The hidden layers are the most important part of a CNN, and the network's performance can be fine-tuned by adjusting the number of hidden layers and the quantity of filters in each layer. A common CNN structure involves numerous convolutional layers, followed by one or more pooling layers, and then a fully connected layer that generates the final output.

12.3.2.1 Convolutional Neural Network Layers

A full Convolutional Neural Network architecture is often called "convnets." Convnets consist of a sequence of layers, with each layer transforming one volume into another using a differentiable function [24]. The layers of a Convolutional Neural Network (CNN) can be broadly categorized into the following groups:

12.3.2.1.1 Convolutional Layer

The main purpose of this layer is to extract features from the input image. It accomplishes this by conducting a convolution operation on the input image, utilizing a filter or kernel to identify and extract certain features.

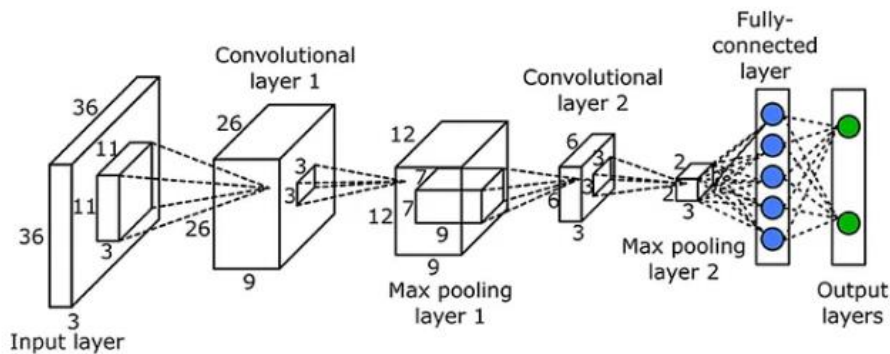


Figure 19: Convolutional Layer [23]

The convolution operation involves sliding the kernel over the input image, multiplying corresponding elements, and summing the results to produce the output. This operation can be expressed as follows:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) \cdot K(m, n) \quad (2)$$

Where;

I – Input image

K – Convolutional Kernel (Filter)

i, j – Indices of the output feature map

m, n – Indices of the filter

In this equation, the convolutional kernel K is a 3D filter with dimensions $m \times n \times c$, where c corresponds to the number of channels in the RGB image. The sum is taken over all spatial dimensions and channels.

12.3.2.1.2 Pooling Layers

Pooling layers reduce the dimensionality of the feature maps produced by the convolutional layers. This is done by down-sampling the feature maps using a pooling operation, such as max or average pooling. This is where features are extracted from images. Pooling layers consolidate the features learned by CNNs.

There are two types of pooling layers that are used:

- **Max Pooling:** This pooling operation identifies the maximum element within the filter-covered region of the feature map. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map. Max Pooling retains the most prominent features of the feature map, and the returned image is sharper than the original image. Stride is a parameter that controls how the filter moves over the input image in a convolutional neural network (CNN). A filter is a small matrix of weights applied to a CNN's input image. The filter learns to detect specific spatial features in the image, such as edges, corners, and shapes. The size of the filter determines the size of the spatial features it can detect. A larger filter will be able to detect larger spatial features, while a smaller filter will be able to detect smaller spatial features. The stride of a filter affects the size of the output feature map. A larger stride will result in a smaller output feature map. This is because the filter will move over the input image fewer times, and each time it moves, it will only process a smaller portion of the input image. Figure 20 below illustrates the max pooling using (2x2) filter and stride of (2, 2).

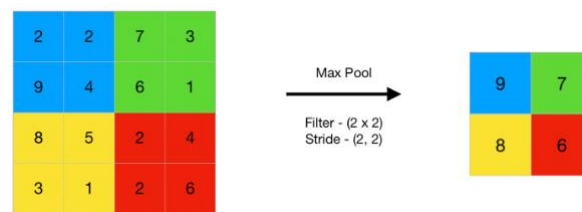


Figure 20: Max Pooling [21]

- **Average Pooling:** Average pooling computes the mean of the elements within the filter-covered region of the feature map. Average pooling provides the average of the features found within that patch. It improves the image's smoothness while keeping the core features of the image [24].

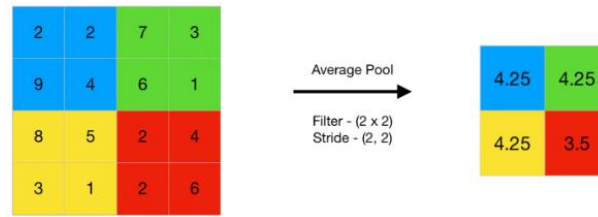


Figure 21: Average Pooling [21]

12.3.2.1.3 Flattening

Following the convolution and pooling layers, the resulting feature maps are turned into a one-dimensional vector. This flattening phase prepares these vectors for transmission to a fully connected layer for classification [21].

12.3.2.1.4 Activation Layer

The activation layer takes the output of the pooling layer and applies a non-linear activation function, such as the ReLU (Rectified Linear Unit) function. The goal of this function is to incorporate non-linearity into the model, allowing it to learn more complicated representations of the input data.

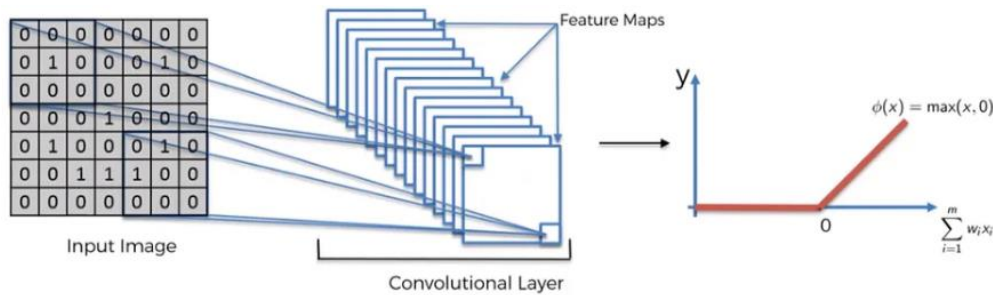


Figure 22: Activation Layer [23]

12.3.2.1.5 Fully Connected Layer

The fully connected layer is a conventional neural network layer that connects all neurons in the preceding layer to all neurons in the following layer. The role of this layer is to combine the features gathered by convolutional and pooling layers to make the prediction.

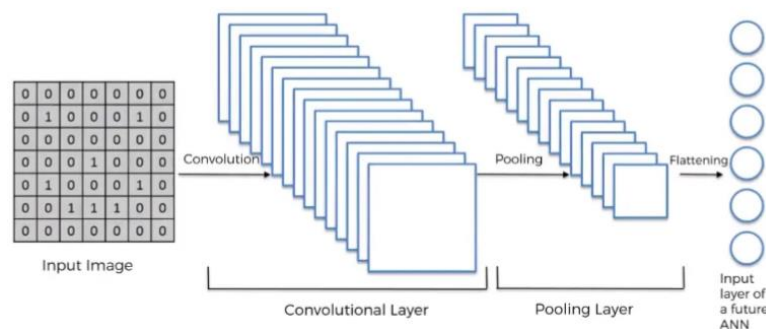


Figure 23: Fully Connected Layer [23]

12.3.2.1.6 Normalization Layer

The normalization layer performs normalizing procedures such as batch normalization and layer normalization. This procedure aims to keep well-conditioned activations within each layer while avoiding overfitting.

12.3.2.1.7 Dropout Layer

The dropout layer serves the purpose of mitigating overfitting by introducing random dropout of neurons during training. This mechanism safeguards against the model memorizing the training data and encourages generalization to new, unseen data. The output of this layer is expressed as follows:

$$Output = \frac{Input}{1 - Dropout Rate} \quad (3)$$

12.3.2.1.8 Dense Layer

Following the feature extraction process by the convolutional and pooling layers, the dense layer comes into play to merge these features and provide the final prediction. In CNNs, the dense layer is typically the final layer responsible for generating output predictions. The activations from previous layers are flattened and transmitted as inputs to the dense layer. The dense layer computes the dot product of the input (x) and the weight matrix (W), adds a bias term (b), and applies the activation function (σ).

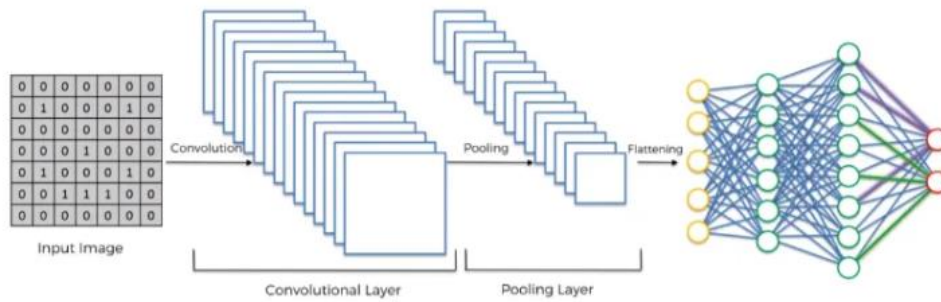


Figure 24: Dense Layer [23]

This function can be expressed using the following equation:

$$f(x) = \sigma(W \cdot x + b) \quad (4)$$

Where;

W – Weight Matrix

x – Input Vector

b – Bias Vector

σ – Activation Function

12.4 CNN Classifier - Model Design and Implementation

The heart of the diabetic retinopathy detection system is the Convolutional Neural Network (CNN). The architecture uses the TensorFlow Keras framework and follows a sequential structure.

Figure 25 below shows the flowchart for designing the DR classification algorithm.

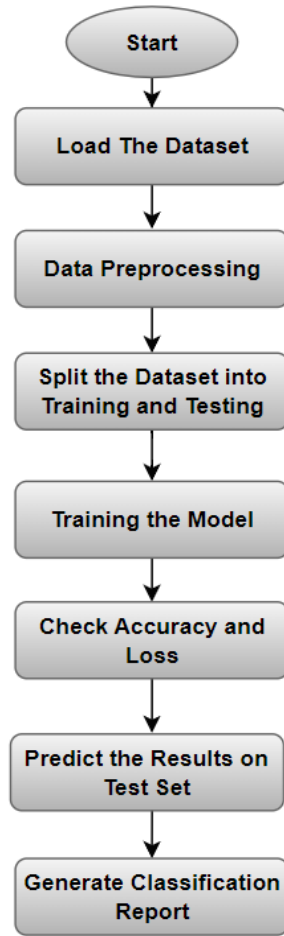


Figure 25: Model Development Flow Chart

The model is as follows:

- **Input Layer (*Resize_Rescale*):** The input image, initially in the shape (224, 224, 3) for width, height, and RGB channels, is first normalized to a range of [0, 1]. This is done by dividing each pixel value by 255, assuming the original pixel values are in the range [0, 255]. The normalized image is then resized to the fixed dimensions of 224x224, making it suitable for further processing.
- **Convolutional Block 1:**
 1. **Conv2D (64 filters, kernel size 3x3, activation='relu', input_shape=(224, 224, 3))**
 - This convolutional layer applies 64 filters of size 3x3 to the input image.
 - The activation function used in this block is ReLU. This function introduces non-linearity by setting all negative values to zero, helping the model learn complex patterns. If the function receives any negative input, it returns 0; however, if the function receives any positive value x , it returns that value. This function can therefore be expressed as follows:
$$f(x) = \max(0, x) \quad (5)$$

Where;
 x – input value
 - The output shape is calculated using the following formulas [25]:

$$Output\ height = \frac{input\ height - kernel\ size + 2 \times padding}{stride} + 1 \quad (6)$$

Where;

Output height - is the resulting height of an image

Input height – is the initial height of an image which is 224

kernel size – 3

padding – since there's no padding done this term is equal to 0

stride – 1

$$Output\ width = \frac{input\ width - kernel\ size + 2 \times padding}{stride} + 1 \quad (7)$$

Where;

Output width – is the resulting width of an image

Input width – is the initial width of an image which is 224

- For the first layer, since there is no padding applied and stride is 1, the output shape becomes (222, 222, 64)

2. Conv2D (64 filters, kernel size 3x3, activation='relu')

- Another convolutional layer with similar parameters is applied, resulting in the output shape (220, 220, 64).

3. MaxPooling2D (2x2)

- MaxPooling downsamples the spatial dimensions by selecting the maximum value in each 2x2 local region.
- The output shape is calculated using the following formulas [25]:

$$Output\ height = \left\lfloor \left(\frac{input\ height - P}{stride} \right) \right\rfloor + 1 \quad (8)$$

Where;

Input height – 220

P – Pooling window size which is 2

stride – 1

$$Output\ width = \left\lfloor \left(\frac{input\ width - P}{stride} \right) \right\rfloor + 1 \quad (9)$$

Where;

Input width – 220

- The output shape after pooling becomes (110, 110, 64)

• Convolutional Block 2:

1. Conv2D (128 filters, kernel size 3x3, activation='relu')

- This convolutional layer with increased filters extracts more complex features, resulting in an output shape of (108, 108, 128).

2. Conv2D (128 filters, kernel size 3x3, activation='relu')

- Another convolutional layer with similar parameters yields an output shape of (106, 106, 128).

3. MaxPooling2D (2x2)

- MaxPooling reduces spatial dimensions, resulting in an output shape of (53, 53, 128).
- **Convolutional Block 3:**
 1. **Conv2D (64 filters, kernel size 3x3, activation='relu')**
 - A convolutional layer with 64 filters is applied, resulting in an output shape of (51, 51, 64).
 2. **Conv2D (64 filters, kernel size 3x3, activation='relu')**
 - Another convolutional layer with similar parameters produces an output shape of (49, 49, 64).
 3. **MaxPooling2D (2x2)**
 - MaxPooling reduces spatial dimensions, resulting in an output shape of (24, 24, 64).
- **Convolutional Block 4:**
 1. **Conv2D (32 filters, kernel size 3x3, activation='relu')**
 - A convolutional layer with 32 filters is applied, resulting in an output shape of (22, 22, 32).
 2. **Conv2D (32 filters, kernel size 3x3, activation='relu')**
 - Another convolutional layer with similar parameters produces an output shape of (20, 20, 32).
 3. **MaxPooling2D (2x2)**
 - MaxPooling reduces spatial dimensions, resulting in an output shape of (10, 10, 32).
- **Flattening:**
 1. This layer has no parameters and has a sole purpose of transforming the 3D output into a 1D array. The resulting output can be calculated using the following formula:
$$Output = width \times height \times No. RGB channels \quad (10)$$

Where;
No. RGB channels – 3
width – 10
height – 10
 2. The resulting output is 3200.
- **Fully Connected Layers:**
 1. **Dense (256 neurons, activation='relu')**
 - A fully connected layer with 256 neurons introduces non-linearity.
 - The output shape becomes 256.
 2. **Dense (128 neurons, activation='relu')**
 - Another fully connected layer with 128 neurons refines features.
 - The output shape becomes 128.
- **Dropout Layer (50%)**
 1. Dropout randomly sets half of the input units to zero during training, preventing overfitting.
- **Output Layer:**
 1. **Dense (5 neurons, activation='softmax')**
 - The output layer with softmax activation produces probabilities for each of the five classes. Softmax activation function is expressed as follows:
$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^C e^{x_j}} \quad (11)$$

Where;
 x_i - The raw score (logit) for the i -th class. It represents the unnormalized output of the neural network for the i -th class.

e^{x_i} - The exponential of the raw score for the i -th class. This transformation helps amplify the differences between scores and ensures non-negative values.

$\sum_{j=1}^C e^{x_j}$ - The sum of the exponentials of raw scores across all C classes. It normalizes the scores and ensures that the resulting probabilities sum to 1.

$Softmax(x_i)$ - The softmax-transformed probability for the i -th class. It represents the likelihood of the input belonging to the i -th class.

- The final output shape is 5 representing the predicted probabilities for each class.

The deep learning model designed above, particularly the architecture of convolutional block 1 and 2 strongly draws inspiration from work of Nagaraj et al. [8]. The fully connected and dropout layer structure is motivated by the deep learning model proposed by Nasir et al. [7]. The convolutional blocks 3 and 4 are designed and added as part of the design choice with the aim to enhance model's performance. The implementation of the model is shown below.

```
model = models.Sequential([
    resize_rescale,
    # Convolutional Block 1
    layers.Conv2D(64, (3, 3), 1, activation='relu', input_shape=(IMAGE_SIZE,
    IMAGE_SIZE, 3)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(),

    # Convolutional Block 2
    layers.Conv2D(128, (3, 3), 1, activation='relu'),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(),

    # Convolutional Block 3
    layers.Conv2D(64, (3, 3), 1, activation='relu'),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(),

    # Convolutional Block 4
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D(),

    # Flatten and Fully Connected Layers
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dense(128, activation='relu'),

    # Dropout Layer
    layers.Dropout(0.5),

    # Output Layer
    layers.Dense(5, activation='softmax')
])
model.build(input_shape=(32, IMAGE_SIZE, IMAGE_SIZE, 3))
```

Table 6 below shows the architecture of the detection method. It has 18 layers, 8 of which are convolutional.

Table 6: Proposed CNN Model

Layer (Type)	Output Shape	Param #
Sequential (Sequential)	(None, 224, 224, 3)	0
conv2d_127 (Conv2D)	(32, 222, 222, 64)	1792
conv2d_128 (Conv2D)	(32, 220, 220, 64)	36928
max_pooling2d_110 (MaxPooling)	(32, 110, 110, 64)	0
conv2d_129 (Conv2D)	(32, 108, 108, 128)	73856
conv2d_130 (Conv2D)	(32, 106, 106, 128)	147584
max_pooling2d_111 (MaxPooling)	(32, 53, 53, 128)	0
conv2d_131 (Conv2D)	(32, 51, 51, 64)	73792
conv2d_132 (Conv2D)	(32, 49, 49, 64)	36928
max_pooling2d_112 (MaxPooling)	(32, 24, 24, 64)	0
conv2d_133 (Conv2D)	(32, 22, 22, 32)	18464
conv2d_134 (Conv2D)	(32, 20, 20, 32)	9248
max_pooling2d_113 (MaxPooling)	(32, 10, 10, 32)	0
flatten_21 (Flatten)	(32, 3200)	0
dense_47 (Dense)	(32, 256)	819456
dense_48 (Dense)	(32, 128)	32896
dropout_18 (Dropout)	(32, 128)	0
dense_49 (Dense)	(32, 5)	645

The architecture of the deep learning model designed and implemented is represented in Figure 26 below. Each component is color-coded for clarity: convolutional layers in light blue, max pooling layers in light red, the flatten layer in light yellow, the dropout layer in light grey, the input layer in light green, and the dense layers in light purple. The final dense layer is the output layer with Softmax activation function with 5 neurons corresponding to five possible DR classes.

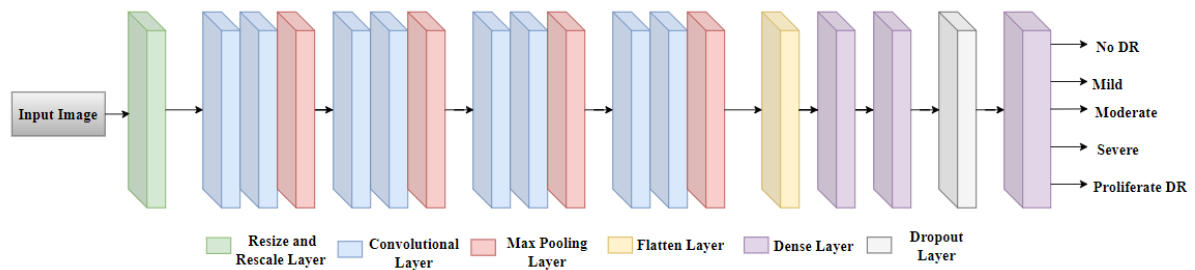


Figure 26: Architecture of Classifier

The summary of the model is as follows:

- **Total Params:** This indicates how many parameters there are in the model in overall. The weights and biases the model employs in its prediction process are called parameters. There are 1,251,589 parameters in the built model.
- **Trainable Params:** These are parameters that are updated during the training process. The model learns and adjusts these parameters to minimize the loss function. The developed model consists of 1,251,589 trainable parameters.
- **Non-Trainable Params:** Throughout the training process, these parameters are not changed. Usually, they are connected to processes or layers without learnable parameters. The developed model does not have any non-trainable parameters as all 1,251,589 parameters are trainable.

12.4.1.1 Model Compilation

Once the model architecture is defined, it is essential to specify its compilation parameters. This step involves configuring important aspects of the model training process, including the choice of optimizer, the loss function, and performance metrics.

- **Optimizer:** The model is compiled using the **Adam optimizer**. Adam (short for Adaptive Moment Estimation) is a popular optimization algorithm that individually adapts learning rates for each model parameter. It combines the advantages of it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithms, making it effective for a wide range of deep learning tasks.

Momentum:

This algorithm is employed to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Leveraging these averages accelerates the convergence of the algorithm towards minima [21].

$$w_{t+1} = w_t - \alpha m_t \quad (12)$$

Where;

$$m_t = \beta m_{t-1} + (1 - \beta) \left[\frac{\partial L}{\partial w_t} \right] \quad (13)$$

Where;

m_t - aggregate of gradients at time t [current] (initially, $m_t = 0$)

m_{t-1} - aggregate of gradients at time t-1 [previous]

w_t - weights at time t

w_{t+1} - weights at time t+1

α_t - learning rate at time t

∂L - derivative of Loss Function

∂w_t - derivative of weights at time t

B - Moving average parameter (const, 0.9)

Root Mean Square Propagation (RMSP):

Root Mean Square Prop (RMSprop) serves as an adaptive learning algorithm designed to improve AdaGrad. Unlike AdaGrad, which accumulates the sum of squared gradients, RMSprop employs the 'exponential moving average' approach.

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t + \epsilon)^{\frac{1}{2}}} * \left[\frac{\partial L}{\partial w_t} \right] \quad (14)$$

Where;

$$v_t = \beta v_{t-1} + (1 - \beta) \left[\frac{\partial L}{\partial w_t} \right]^2 \quad (15)$$

v_t - sum of square of past gradients.

ϵ - A small positive constant 10^{-8}

- **Loss Function:** The Sparse Categorical Cross entropy has been selected as the loss function for this model. Multi-class classification problems, like grading different forms of diabetic retinopathy, are a good fit for this loss function. It quantifies the dissimilarity between the true class labels and the predicted class probabilities, with 'from_logits' set to 'False'. This indicates that the predicted probabilities are presented as normalized probabilities rather than as raw logits. Sparse Categorical Crossentropy is expressed as follows:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C y_{i,j} \cdot \log(\hat{y}_{i,j}) \quad (16)$$

Where;

N - number of samples in the dataset

C - number of classes

$y_{i,j}$ - binary indicator of whether class j is the true class for sample i .

$\hat{y}_{i,j}$ - predicted probability that sample i belongs to class j

- **Metrics:** Accuracy is the selected metric to track the model's performance during training and evaluation. Accuracy represents the proportion of correctly classified samples from the total samples in the evaluation dataset. It offers a simple way to assess how well the model performs in classifying patients with varying degrees of diabetic retinopathy.

12.4.1.2 Model Training

This section delves into the training and evaluation processes applied in the development of the diabetic retinopathy detection system. It outlines the procedures, the key evaluation metrics, and the performance criteria used to assess the model's effectiveness.

12.4.1.2.1 Training Process

The development's main phase is training the Convolutional Neural Network Model, tasked with classifying retinal images into distinct diabetic retinopathy grades.

- **Data Loading in Batches:** The retinal images are loaded and analysed in 40-image batches throughout the training phase. This data loading and processing technique offers several advantages, including improved learning, quicker convergence, and less memory utilization.
- **Epochs:** An epoch is a single complete pass through the entire training dataset. It represents the summation of all iterations necessary to process and train the deep learning model on the complete training dataset within one cycle [26]. The CNN model for this system is trained over 50 epochs on APTOS2019 dataset since this dataset is small, and trained over 30 epochs on the EyePACS dataset as it is very large. The decisions behind the number of epochs were made in consideration of the available computational resources.

12.4.1.3 Model Evaluation

The evaluation phase is a critical step in the development of a CNN model for retinal image classification. It involves assessing the model's performance, reliability, and precision on a held-out test set. The implemented is trained and evaluated on both datasets, independently. This is done to assess performance of the model under different data conditions. The goal is to understand whether the model's performance improves with more diverse and extensive training data.

12.4.1.3.1 Evaluation Metrics

Several key evaluation metrics are employed to measure the performance of the diabetic retinopathy classification model [27]:

- **Accuracy:** An elemental metric that computes correctly classified images' ratios to the total number of images in the testing set.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (17)$$

Where;

TP – True Positives

FP – False Positives

FN – False Negatives

TN – True Negatives

- **Precision and Recall:** Precision assesses the proportion of true positive predictions among all positive predictions, while recall calculates the ratio of true positive predictions to all actual positive samples.

$$Precision = \frac{TP}{TP + FP} \quad (18)$$

$$Recall(Sensitivity) = \frac{TP}{TP + FN} \quad (19)$$

- **F1-Score:** The F1-Score is the harmonic mean of precision and recall, offering a balanced measurement of the model's performance, especially in imbalanced class distribution cases.

$$F1\ Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (20)$$

- **Confusion Matrix:** The confusion matrix provides a granular breakdown of true positives, true negatives, false positives, and false negatives. This detailed analysis aids in identifying specific areas where the model excels or requires further improvement. Confusion matrix (also known as “Contingency Matrix”) gives a good overview of the performance of a classifier. Table 7 below shows the typical structure of the confusion matrix.

Table 7: Confusion Matrix Structure

Actual Class	Predicted Class		
		Positive	Negative
	Positive	TP	FN
	Negative	FP	TN

$$Specificity = \frac{TN}{TN + FP} \quad (21)$$

$$Negative\ predictive\ value\ (NPV) = \frac{TN}{TN + FN} \quad (22)$$

$$False\ negative\ rate\ (FNR) = 1 - Recall \quad (23)$$

$$False\ positive\ rate\ (FPR) = 1 - Specificity \quad (24)$$

$$False\ discovery\ rate\ (FDR) = 1 - Precision \quad (25)$$

$$False\ omission\ rate\ (FOR) = 1 - NPV \quad (26)$$

12.4.1.3.2 Model Evaluation on APTOS2019 Dataset

The classification report of the developed model trained on APTOS2019 dataset is shown in Figure 27 below. The model achieved an overall accuracy of 95%. The model achieved 100% precision, recall and f-1 score for all diabetic retinopathy grades except grade 1, moderate diabetic retinopathy grade.

	precision	recall	f1-score	support
0	1.00	0.75	0.86	4
1	0.67	1.00	0.80	2
2	1.00	1.00	1.00	5
3	1.00	1.00	1.00	7
4	1.00	1.00	1.00	4
accuracy			0.95	22
macro avg	0.93	0.95	0.93	22
weighted avg	0.97	0.95	0.96	22

Figure 27: Classification Report (APTOS2019)

The confusion matrix generated when testing the model using the dataset is shown in Figure 28 below. The model correctly classified all test images. The minor misclassification occurred when the model classified one of the images as mild while the true label is moderate. The total testing dataset consists of 67 images; only one image was misclassified.

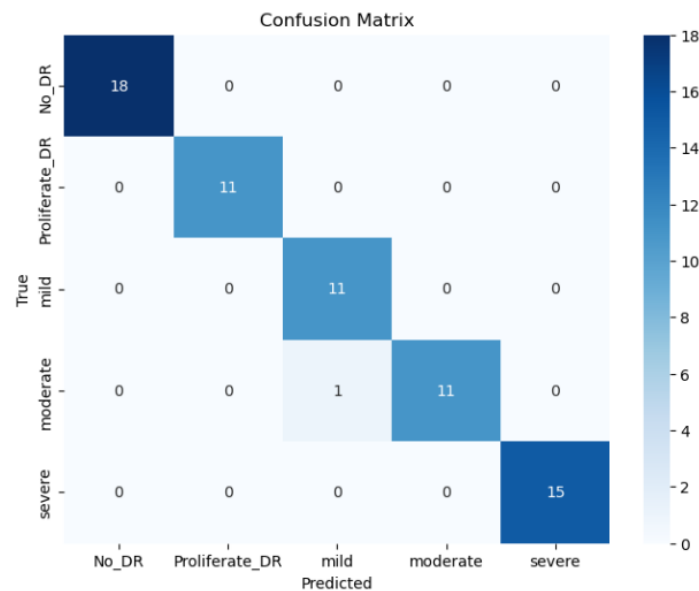


Figure 28: Confusion Matrix (APTOS2019)

12.4.1.3.2.1 Training and Validation Accuracy

Figure 29 presents the training and validation accuracy curves plotted over 50 epochs. The training accuracy follows an upward trajectory, demonstrating the model's proficiency in making accurate predictions during training. The validation accuracy also reveals an increasing trend, with some minor fluctuations. These trends confirm the potential of the model to generalize to previously unencountered retinal images.

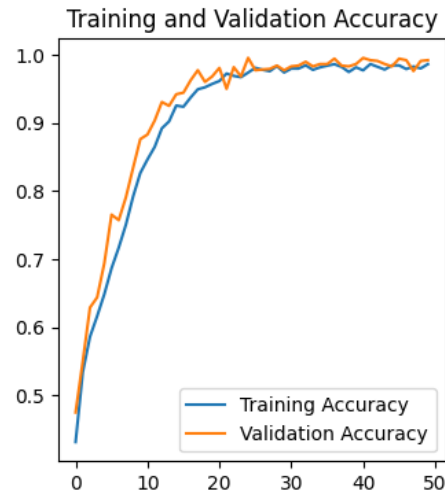


Figure 29: Training and Validation Accuracy (APTOS2019)

12.4.1.3.2.2 Training and Validation Loss

The training and validation loss curves are illustrated in Figure 30 below over 50 epochs. The training loss curve showcases a consistent downward trajectory, indicating that the model progressively minimizes its training loss as it learns from the training dataset. Similarly, the validation loss curve exhibits a decreasing trend with minor fluctuations. This validates the model's capability to generalize to new, unseen data, signifying the avoidance of overfitting.

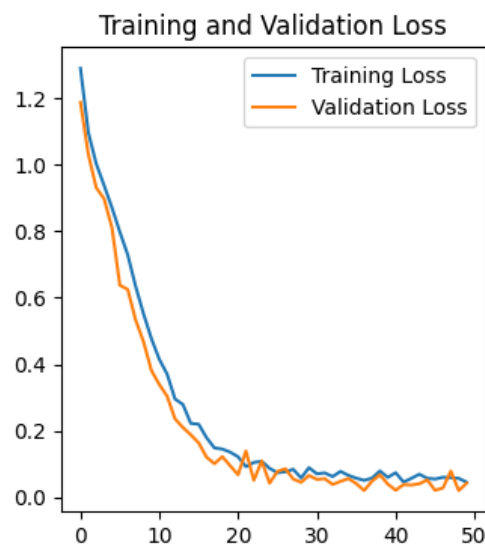


Figure 30: Training and Validation Loss (APTOS2019)

12.4.1.3.2.3 Qualitative Results

Some of the testing results were visualized as shown in Figure 31. The model predicted one of the images as having a proliferate diabetic retinopathy grade, while the actual grade was severe. The model's confidence in this misclassification case was 89.18%, which shows that it was not very certain about its prediction. In other images the model predicted the correct class precisely with minimum confidence of 99.74%.

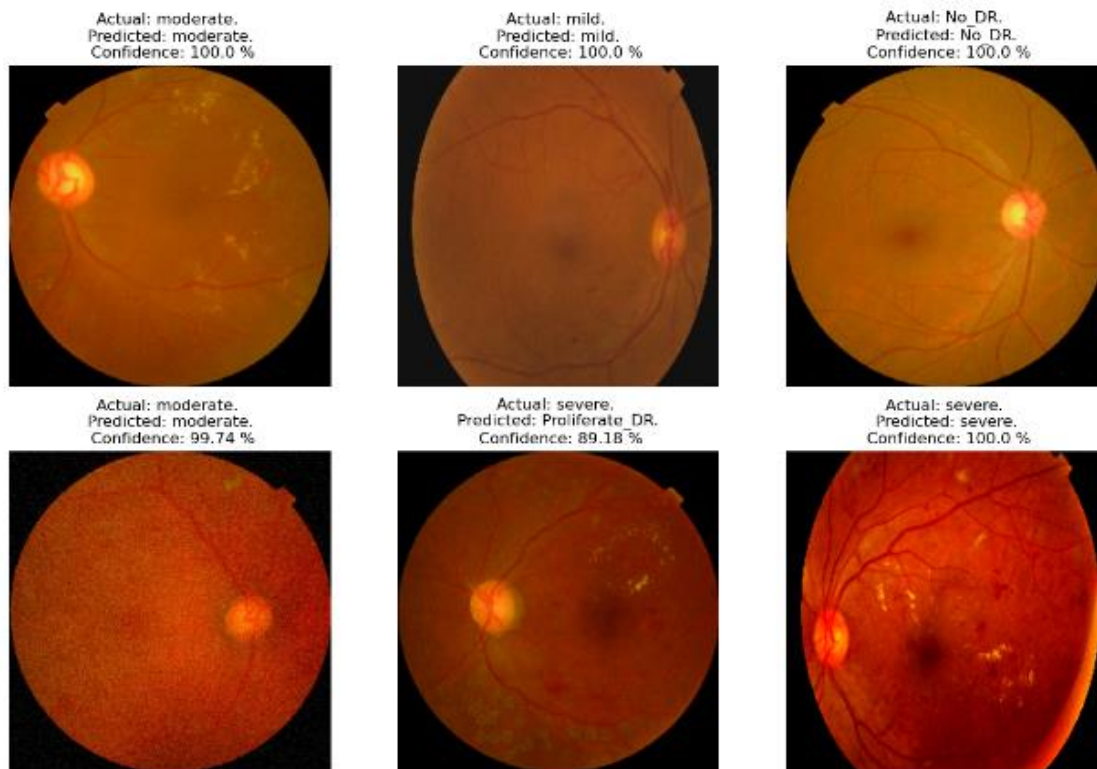


Figure 31: Testing Results Visualization on APTOS2019 Dataset

12.4.1.3.3 Model Evaluation on EyePACS Dataset

The model is trained using the EyePACS larger dataset and the classification report showing the performance of the model is shown in Figure 32 below. The model achieved an overall accuracy of 80%. The model achieved 100% precision, with recall of 89% and f-1 score of 94% for class 3. The model seems to struggle to identify images belonging to class 1 and 2, this is revealed by the precision, recall, and f1-score the model obtained for these classes.

	precision	recall	f1-score	support
0	0.95	0.67	0.78	27
1	0.67	0.67	0.67	24
2	0.63	0.83	0.72	23
3	1.00	0.89	0.94	19
4	0.86	0.94	0.90	32
accuracy			0.80	125
macro avg	0.82	0.80	0.80	125
weighted avg	0.82	0.80	0.80	125

Figure 32: Classification Report (EyePACS)

The confusion matrix generated when testing the model using the dataset is shown in Figure 33 below. It is observed that on Mild class, the model was able to identify 18 images, only one image was labels as Moderate. The model correctly identifies all the available Proliferate DR images and no images was misclassified as Proliferated. In Overall the model struggled to correctly classify some of the images from No DR, Moderate and Severe classes. The total testing dataset consists of 125 images; 25 images were misclassified.

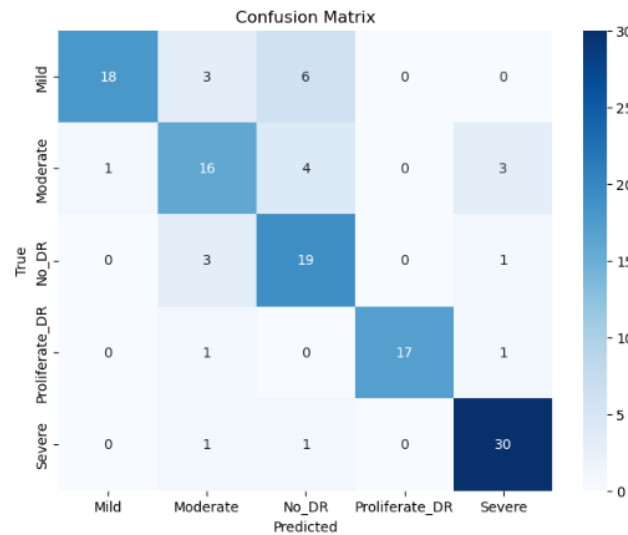


Figure 33: Confusion Matrix (EyePACS)

12.4.1.3.3.1 Training and Validation Accuracy

Figure 34 presents the training and validation accuracy curves plotted over 30 epochs. The upward trend of the training accuracy curve indicates how well the model is learning from the training data. The validation accuracy curve also follows the similar trend that is followed by the training curve, with some minor fluctuations. Ideally, both curves exhibit a rising trend, indicating effective learning and generalization.

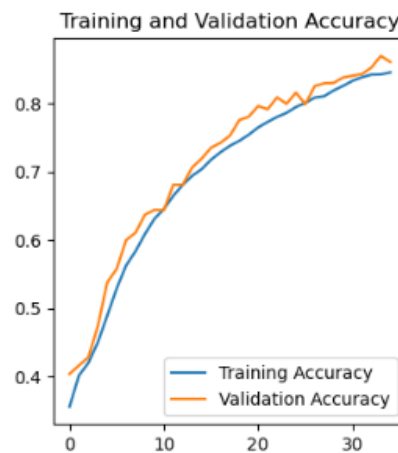


Figure 34: Training and Validation Accuracy (EyePACS)

12.4.1.3.3.2 Training and Validation Loss

The training and validation loss curves are illustrated in Figure 35 below over 30 epochs. The training loss curve showcases the model's progression in minimizing the error on the training dataset as training epochs advance. A consistent decreasing trend is observed in training loss curve, illustrating the model's ability to generalize to new, unseen data. Similarly, the validation loss curve follows a decreasing trend with minor fluctuations. This validates the model's capability to generalize to new, unseen data, signifying the avoidance of overfitting. A consistent decrease in both curves reflects effective learning and generalization. Discrepancies such as widening gap between the two curves and unexpected rise in the validation loss indicate potential overfitting issue. However, both curves steadily decrease indicating convergence and capability of the model to capture meaningful patterns and features without memorizing the training set.



Figure 35: Training and Validation Loss (EyePACS)

12.4.1.3.3.3 Qualitative Results

Some of the testing results were visualized as shown in Figure 31. The model correctly predicted 4 out of 6 images shown. The misclassifications occurred are accompanied by low confidence level of the model. In this case the model predicted the third image on the top row as having Proliferate DR while the true label is Moderate, the confidence of the model in this case is 64.19%. The last misclassification occurred when the model predicted the third image in the last row as having Mild DR while the true label is Severe, the confidence of the model in this case 52.7%, indicating high uncertainty in the prediction made.

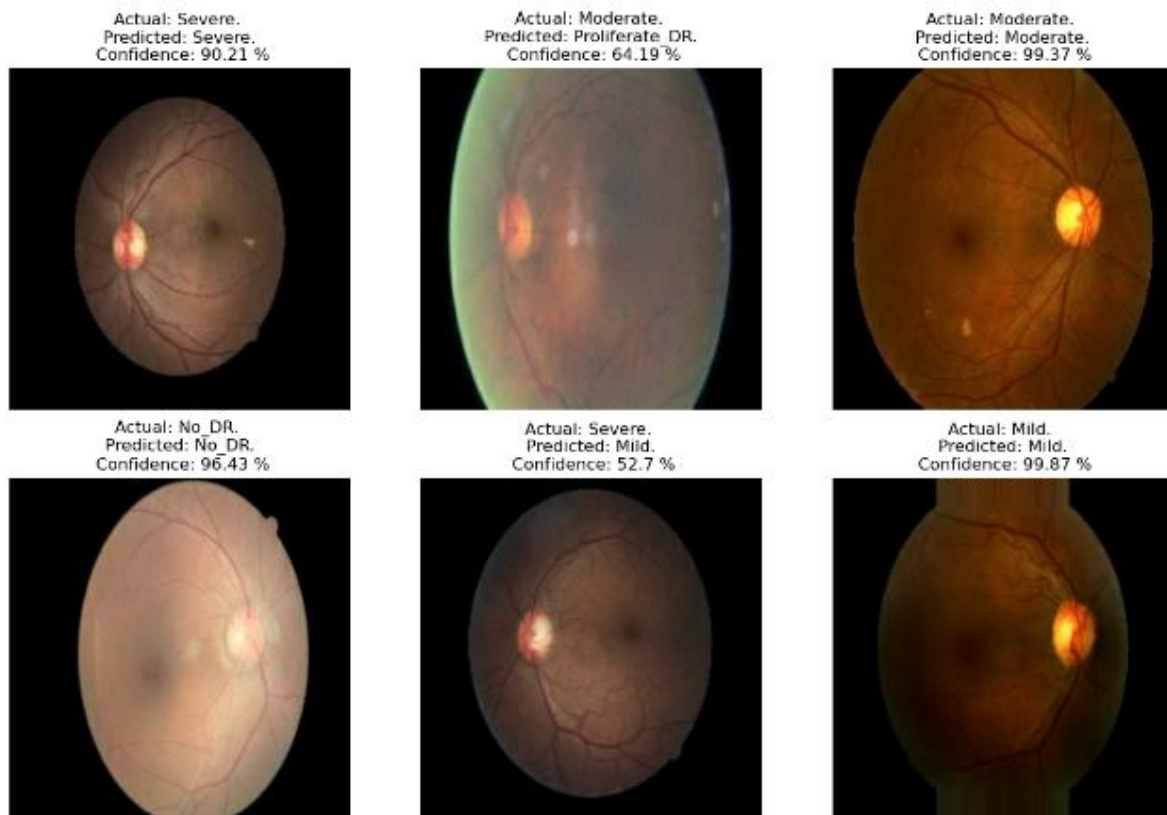


Figure 36: Testing Results Visualization on EyePACS Dataset

12.5 Results Analysis and Discussion

This section compares the performance of the trained deep learning model proposed with state of the art methods trained on either or both APTOS2019 and EyePACS dataset. The comparison is summarized in Table 8 and 9 below and the evaluation metric taken into consideration is the accuracy.

Table 8: Comparative Analysis of results achieved by the proposed method with state-of-the-art using APTOS Dataset

Reference	Authors	Methods	Data Augmentation Used	Accuracy
[28]	Ishaq et. al	Transfer-Learning: VGG16, VGG19, Xception, InceptionV3, and MobileNetV2	Yes	86%
[29]	Omar et. al	Features concatenation (ResNet50, EfficientNetB5, InceptionV3)	Yes	97.24%
[1]	Manasi et. al	ResNet-50	Yes	71.51%
[30]	Shital Firke and Ranjan Jain	CNN	Yes	96.15%
[31]	Nandhini et. al	DiaNet	Yes	90.02%
Proposed	---	CNN	Yes	95%

Table 9: Comparative Analysis of results achieved by the proposed method with the state-of-the-art using EyePACS Dataset

Reference	Authors	Methods	Data Augmentation used	Accuracy
[32]	Mohamed Cheteou and Moulay Akhloufi	DCNN Inception-Resnet-V2	Yes	97.9%
[33]	Ziyuan et. al	BiRA-Net	Yes	54.31%
[34]	Peisheng et. al	TSBN (ResNet-50)	Yes	52.12%
		TSBN (BiRA-Net)		55.13%
[35]	Wenhui et. al	NNMobile-Net	Yes	94.4%
[36]	Gangwar et. al	Proposed Model	Yes	72.33%
		GoogleNet		66.03%
Proposed	---	CNN	Yes	80%

Table 8 and 9 clearly show that the developed model achieved a noteworthy achievement in accuracy when trained with APTOS and EyePACS datasets, respectively. The most notable achievement is the accuracy obtained on the APTOS dataset, the model outperforms some of the state-of-the-art models with 95% accuracy compared to the range of 71.51% to 97% achieved by other models. This

performance was expected as the APTOS dataset is small. Furthermore, when the model trained on the larger dataset (EyePACS), the model maintains a competitive accuracy of 80%, positioning itself favorably against other models with accuracies ranging from 54.3% to 97%. The performance drop was expected as the size of the dataset increased and the computational resources were limited, which led to the decision to train the model on 30 epochs instead of 50. Therefore, it was noted that the size of the dataset plays a huge role in model training, and the developed model's ability to achieve robust accuracy on both datasets underscores its adaptability and effectiveness across varying sizes of data. These results affirm the effectiveness of the developed model in diabetic retinopathy detection, showcasing its potential for real-world applications.

12.6 User Interface Design

The designed user interface for the diabetic retinopathy detection system is illustrated in Figure 37 below. The interface has been designed to be easy for the user to ensure a seamless user experience.

- **Upload Button:** The user uploads a retinal image by clicking the "Upload" button. The system will prompt the user to select the retina image to upload for analysis. Upon successful upload of image, the analysis process is then initiated.
- **Clear Button:** A clear button is provided to allow the user to remove the selected image should they wish to replace it with another one image or restart the analysis process.
- **Prediction Label:** The predicted diabetic retinopathy grade is prominently displayed on the interface. This label indicates the system's diagnosis based on the analysis of the uploaded image.
- **Confidence Score:** The system provides a confidence score alongside the prediction label. This score informs users about the level of certainty associated with the diagnostic result.



Figure 37: User Interface

12.7 Web Application Development

This section provides an overview of the development process of the web application used as part of the diabetic retinopathy detection system. This section explores the layout, colour schemes, and interactive elements implemented to enhance the user interface and simple navigation. Furthermore, the section explains in detail the interaction between frontend and the backend units of the system.

12.7.1 Frontend Development

A combination of HTML, CSS, JavaScripts and React was used to implement the user interface. In the frontend, React components are organized to create a structured and maintainable codebase.

12.7.2 Code Structure

The source code is contained in the src folder. This folder has the following files:

12.7.2.1 App.js

- The main entry point of the React application.
- Imports and renders the **ImageUpload** component.

12.7.2.2 App.test.js

- Contains a test case for rendering the **App** component.

12.7.2.3 Home.js

- Implements the ImageUpload component.
- Utilizes Material-UI components for UI elements.
- Handles image upload, processing, and displays results.
- Utilizes Axios for making HTTP requests to the backend.

12.7.2.4 Index.js

- Renders the **App** component and connects it to the root HTML element.

12.7.2.5 Index.css

- Contains styling rules for the entire application.

12.7.3 Components

- **ImageUpload Component:**
- Handles image upload, preview, and processing.
- Displays the processed results, including DR grade and confidence.
- Implements Material-UI components for styling.

12.7.4 Axios

- Handles HTTP requests to the backend.

12.7.5 Styles

- **Material-UI Styling:**
- Utilizes Material-UI's makeStyles and withStyles for styling components.
- Material-UI styling is used for design and development of the responsive user-interface.

12.7.5.1 Functionality Features

The web application encompasses various features, including image upload, processing, and displaying diagnostic results. Each feature is designed to contribute to the overall diagnostic capabilities of the system, ensuring that healthcare professionals can efficiently utilize the application for diabetic retinopathy detection.

12.7.5.2 Main Page

The main page of the Diabetic Retinopathy Detection System serves as the user's gateway to seamless image diagnosis. Users are prompted to effortlessly upload a retina image by simply dragging and dropping it onto the designated area or clicking the upload area to be directed to the files of the system and selected the retina image. Figure 38 below shows the main page of the system before the retina image is uploaded for analysis.

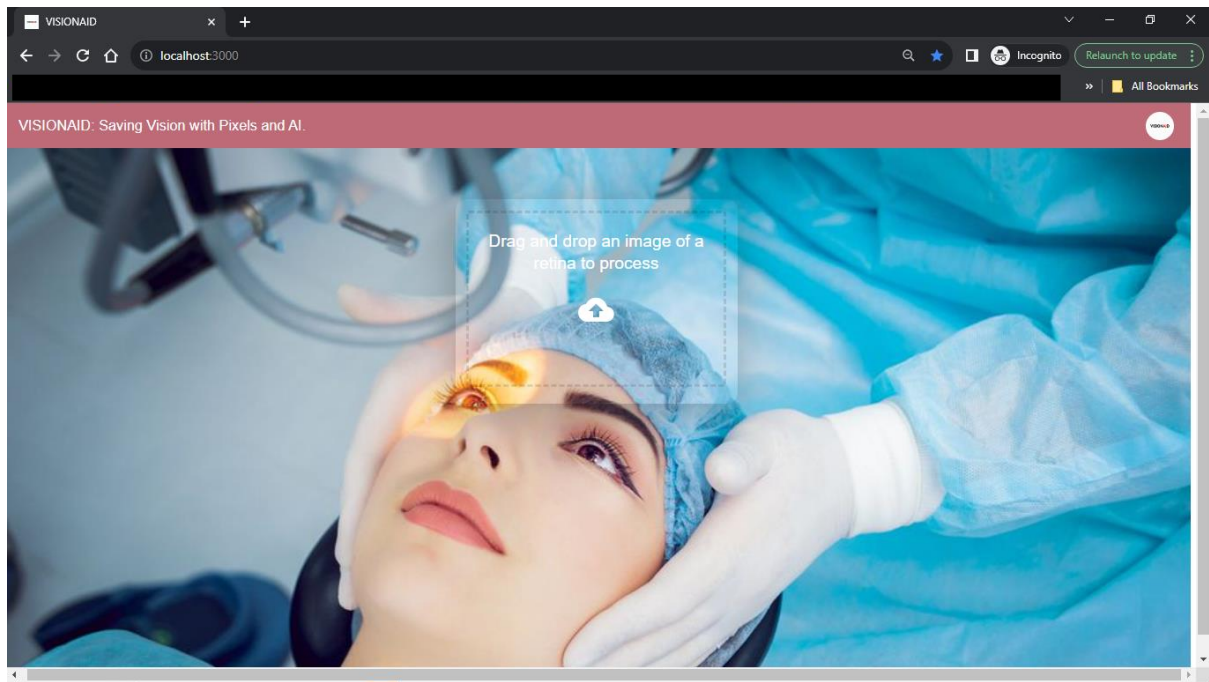


Figure 38: Main Page of the System

For added convenience, a "Clear" button is provided to remove any uploaded image, allowing the healthcare professionals to refine their selections. Upon image upload, the system processes the retina image and promptly returns the Diabetic Retinopathy (DR) grade along with a corresponding confidence level. This streamlined interface ensures a user-friendly experience, enabling healthcare professionals to efficiently obtain diagnostic results with minimal effort. Figure 39 below shows an example diagnostic results (No DR) with confidence level of 99.99%.

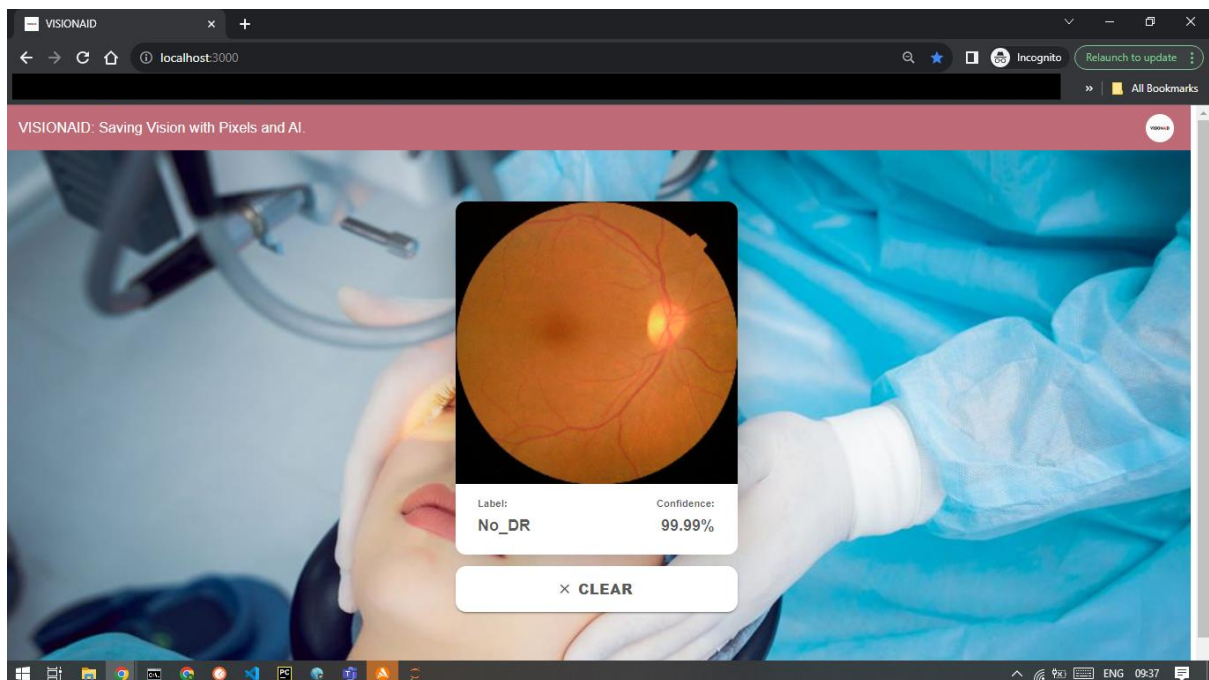


Figure 39: Diagnostic Results Example (No DR)

12.7.6 Server Implementation (Backend)

The backend development leverages FASTAPI to handle data processing, storage, and management. The server side of the system consists of the following key components:

12.7.6.1 App Creation

- Creates a FastAPI instance named app.
- Defines CORS (Cross-Origin Resource Sharing) middleware to allow requests from specified origins.

12.7.6.2 Model Loading

- Loads a pre-trained deep learning model using TensorFlow (tf.keras.models.load_model).
- The model is used for predicting diabetic retinopathy classes.

12.7.6.3 Endpoint: '/ping'

- Defines a simple endpoint for testing the server's availability.
- Returns a "Hello World" message when accessed.

12.7.6.4 Function: *read_file_as_image*

- Defines a simple endpoint for testing the server's availability.
- Returns a "Hello World" message when accessed.

12.7.6.5 Endpoint: '/predict'

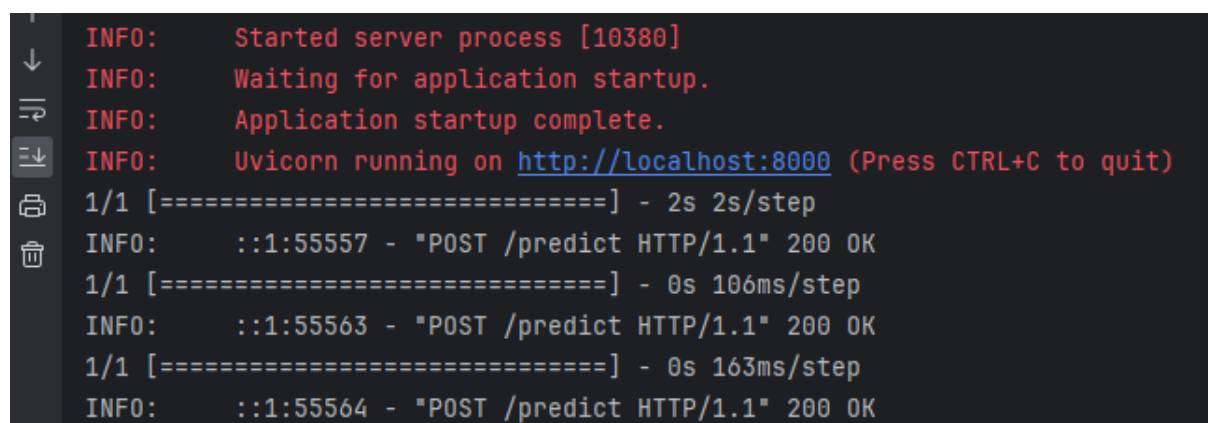
- Accepts an uploaded image file using FastAPI's UploadFile class.
- Utilizes the loaded model to predict the class and confidence of the diabetic retinopathy.
- Returns a JSON response with the predicted class and confidence.

12.7.6.6 Uvicorn.run Statemet

- This statement runs the FastAPI application using the Uvicorn server.
- Specifies the host and port for the server to listen on (localhost:8000).

12.7.6.7 CORS Configuration

- Configures CORS to allow communication with specified frontend origins (localhost and localhost:3000).
- Enables credentials, allows any HTTP method, and permits any header.

A terminal window with a dark background and light-colored text. On the left side, there is a vertical toolbar with icons for back, forward, search, and other navigation functions. The terminal output shows the following lines:

```
INFO:      Started server process [10380]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://localhost:8000 (Press CTRL+C to quit)
1/1 [=====] - 2s 2s/step
INFO:      ::1:55557 - "POST /predict HTTP/1.1" 200 OK
1/1 [=====] - 0s 106ms/step
INFO:      ::1:55563 - "POST /predict HTTP/1.1" 200 OK
1/1 [=====] - 0s 163ms/step
INFO:      ::1:55564 - "POST /predict HTTP/1.1" 200 OK
```

Figure 40: Server Initiation and Response

Figure 40 above illustrates the interaction between the frontend and the Uvicorn server during the diabetic retinopathy detection process. The Uvicorn server, running locally on <http://localhost:8000>, acts as the backend for the application. The interaction between these units is illustrated in the flow chart in Figure 41 below

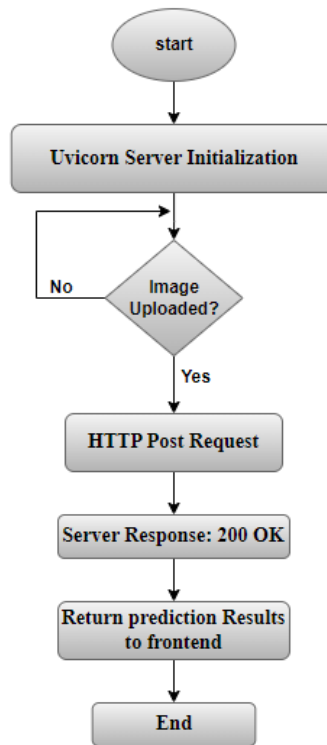


Figure 41: Backend Flow Chart

- a) **Uvicorn Server Initialization**
 - The uvicorn server is initialized when running the server application. This server serves as the backend of the DR detection system.
- b) **Frontend Trigger**
 - The frontend initiates a prediction request by interacting with the Uvicorn server. This request is triggered when a user uploads a retina image through the user interface.
- c) **HTTP Post Request**
 - The communication between the frontend and the server is facilitated through HTTP POST requests. As shown in Figure 40 above, the transmission of a POST request to the server is sent through the ‘/predict’ endpoint on the unicorn server.
- d) **Server Response**
 - The Uvicorn server processes the received POST request, executes the prediction algorithm using the pre-trained model, and generates a response. The "POST /predict HTTP/1.1 200 OK," signifies a successful response from the server.
- e) **Prediction Results**
 - The response from the server includes the predicted class and confidence level.
- f) **Feedback to Frontend**
 - The predicted class and confidence level are then sent back to the frontend, enabling the user interface to display the diagnostic result to the user.

13. Future Work

The designed and implemented diabetic retinopathy system have some room for improvement. The system can be enhanced by exploring transfer learning combined with image segmentation techniques. By leveraging the power of pre-trained models designed for image analysis the system can have a achieve a better performance since these models are trained on very large datasets. Additionally, employment of image segmentation techniques would allow for more detailed analysis, focusing on specific regions of the retina images. Moreover, generating a diagnostic report alongside predictions could offer a comprehensive overview, aiding healthcare professionals in understanding the model's decisions and facilitating communication with patients.

14. Conclusion

Developing an automated diabetic retinopathy detection system using deep learning techniques presents a promising solution to address the challenges associated with manual diagnosis of this critical medical condition. The feasibility study has underscored the project's viability, considering market demand, technical resources, software and hardware capabilities, and implementation feasibility. The developed diabetic retinopathy detection system, integrated with a user-friendly web application, demonstrates a robust and effective solution for early diagnosis. Leveraging a deep learning model trained and evaluated on both the EyePACS and APTOS datasets, the system achieved a notable performance, attaining an accuracy of 80% on the larger EyePACS dataset and an impressive 95% on the smaller APTOS dataset. The comparative analysis against current state-of-the-art models underscores the system's competitiveness in the realm of diabetic retinopathy detection. The successful integration between the web app and the trained model ensures accessibility and ease of use for healthcare professionals, fostering timely and accurate diagnoses in the field of ophthalmology. This project contributes significantly to the ongoing efforts to employ advanced technologies in healthcare for enhanced disease detection and management.

References

- [1] M. Edlabadkar, A. Gotarne, R. Kshirsagar and C. Thorat, "Detection of Diabetic Retinopathy using ResNet50," *Internation Journal of Creative research Thoughts (IJCRT)*, vol. 11, no. 8, pp. 1-7, 2023.
- [2] M. Yadav, R. Goel and R. D, "Deep Learning Based Diabetic Retinopathy Detection from Retina Images," *International Conference on Intelligence Technolgies (CONIT)*, pp. 1-4, 2021.
- [3] N. Zaaboub and A. Douik, "Application of Learning Algorithm for Diabetic Retinopathy Diagnosis," *International Multi-Conference on Systems, Signals & Devices*, pp. 1-5, 2020.
- [4] Harikrishna, Dharsinala, Kumar and N. Udaya, "Artificial Intelligence System for Classification of Diabetic Retinopathy," *Smart Technologies, Communication & Robotics (STCR)*, pp. 1-5, 2022.
- [5] C. Enrique, G. Andres and C. Ricardo, "Detection of Diabetic Retinopathy using SVM," pp. 1-4, 2017.
- [6] M. T. Hagos, S. Kant and S. A. Bala, "Automated Smartphone Based System for Diagnosis of Diabetic Retinopathy," *International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 1-6, 2019.
- [7] N. Nasir, P. Oswald, O. Alshaltone, F. Barneih and M. A. S. Ahmed, "Deep DR: Detection of Diabetic Retinopathy Using," *ASET*, pp. 1-5, 2022.
- [8] G. Nagaraj, C. S. Simha, R. H. C. G and M. D. Indiramma, "Deep Learning Framework for Diabetic Retinopathy Diagnosis," *Proceedings of the Third International Conference on Computing Methodologies and Communication (ICCMC 2019)*, pp. 1-6, 2019.
- [9] V. Vipparthi, D. D. R. Rao, S. Mullu and V. Patlolla, "Diabetic Retinopathy Classification Using Deep Learning Techniques," *Proceedings of the Third International Conference on Electronics and Sustainable Communication Systems (ICESC 2022)*, pp. 1-7, 2022.
- [10] A. Kumar, Arsha and A. Babu, "Diabetic Retinopathy Detection Using Deep Learning Methodology," *IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, pp. 1-6, 2022.
- [11] U. Shrabony and S. Ejaz, "Early Identification of Diabetic Retinopathy Using Deep Learning Model: A Survey," *2022 International Conference on Recent Progresses in Science, Engineering and Technology (ICRPSET)*, pp. 1-4, 2022.
- [12] C. Stedman, "What is data preparation? An in-depth guide to data prep," Tech Accelerator, [Online]. Available: <https://www.techtarget.com/searchbusinessanalytics/definition/data-preparation>. [Accessed 12 September 2023].
- [13] S. R. Rath, "Kaggle," Kaggle, [Online]. Available: <https://www.kaggle.com/datasets/sovit Rath/diabetic-retinopathy-224x224-gaussian-filtered>. [Accessed 14 September 2023].

- [14] F. Chollet, "Preparing the data," in *Deep Learning with Python*, New York, Manning Publications Co., 2018, pp. 92-375.
- [15] D. Matani, "Replace Manual Normalization with Batch Normalization in Vision AI Models," Towards Data Science, 18 May 2023. [Online]. Available: <https://towardsdatascience.com/replace-manual-normalization-with-batch-normalization-in-vision-ai-models-e7782e82193c>. [Accessed 10 September 2023].
- [16] J. Brownlee, "How to Manually Scale Image Pixel Data for Deep Learning," Machine Learning Mastery, 5 July 2019. [Online]. Available: <https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/#:~:text=Normalize%20Pixel%20Values,-For%20most%20image&text=Neural%20networks%20process%20inputs%20using,value%20between%200%20and%201..> [Accessed 1 October 2023].
- [17] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, pp. 1-48, 2019.
- [18] S. Firker and R. Jain, "Convolutional Neural Network for Diabetic Retinopathy Detection," *Proceedings of the International Conference on Artificial Intelligence and Smart Systems (ICAIS-2021)*, no. 549-553, pp. 1-5, 2021.
- [19] V. Nemade, M. Edlabadkar, A. Gotarne, R. Kshirsagar and C. Thorat, "Detection of Diabetic Retinopathy using ResNet50," *INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS(IJCRT)*, vol. 11, pp. 1-7, 2023.
- [20] D. S. Wizards, "A Guide to Data Splitting in Machine Learning," Towards Data Science, 16 November 2022. [Online]. Available: <https://medium.com/@datasciencewizards/a-guide-to-data-splitting-in-machine-learning-49a959c95fa1#:~:text=Data%20splitting%20becomes%20a%20necessary,than%20the%20other%20two%20data..> [Accessed 2 September 2023].
- [21] GeeksforGeeks, "CNN | Introduction to Pooling Layer," GeeksforGeeks, [Online]. Available: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>. [Accessed 5 September 2023].
- [22] Spotfire, "What is a neural network?," Spotfire, [Online]. Available: <https://www.spotfire.com/glossary/what-is-a-neural-network>. [Accessed 3 October 2023].
- [23] N. Shahriar, "What is Convolutional Neural Network — CNN (Deep Learning)," Medium, 1 February 2023. [Online]. Available: <https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5>. [Accessed 14 September 28 2023].
- [24] T. AI, "Introduction To Pooling Layers In CNN," Towards AI, 16 August 2022. [Online]. Available: <https://towardsai.net/p/l/introduction-to-pooling-layers-in-cnn>. [Accessed 16 August 2023].
- [25] Opendgenus, "Calculate output size of Convolution," Opendgenus, [Online]. Available: <https://iq.opendgenus.org/output-size-of-convolution/>. [Accessed 10 November 2023].
- [26] Simplilearn, "What is Epoch in Machine Learning?," Simplilearn, 10 August 2023. [Online]. Available: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-epoch-in-machine->

learning#:~:text=An%20epoch%20is%20when%20all,dataset%20takes%20around%20an%20algorithm.. [Accessed 17 September 2023].

- [27] “Understanding the Classification report through sklearn,” Muthukrishnan, 7 July 2018. [Online]. Available: <https://muthu.co/understanding-the-classification-report-in-sklearn/#:~:text=A%20Classification%20report%20is%20used,classification%20report%20as%20shown%20below..> [Accessed 15 September 2023].
- [28] I. Aiche, Y. Brik, B. Attallah, Z. Zohra and H. Lahmar, “Transfer Learning for Diabetic Retinopathy Detection,” *ternational Conference of Advanced Technology in Electronic and Electrical Engineering (ICATEEE)*, pp. 1-5, 2022.
- [29] O. Boukadoum and N. Benblidia, “Features Extraction and Concatenation of Transfer Learning Models for Early Diagnosis of Diabetic Retinopathy,” *International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS)*, pp. 1-5, 2023.
- [30] S. N. Firke and R. B. Jain, “Convolutional Neural Network for Diabetic Retinopathy Detection,” *nternational Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pp. 1-5, 2021.
- [31] Nandhini, Sowbarnikkaa, Mageshwari and Saraswathy, “An Automated Detection and Multi-stage classification of Diabetic Retinopathy using Convolutional Neural Networks,” *International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*, pp. 1-6, 2023.
- [32] M. Chetoui and M. Akhloufi, “Explainable end-to-end deep learning for diabetic retinopathy detection across multiple datasets,” *Society of Photo-Optical Instrumentation Engineers (SPIE)*, pp. 1-25, 2020.
- [33] Z. Zhao, K. Zhang, X. Hao, J. Tian, M. C. H. Chua, L. Chen and X. Xu, “BIRA-NET: BILINEAR ATTENTION NET FOR DIABETIC RETINOPATHY GRADING,” *Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System*, pp. 1-5, 2019.
- [34] P. Qian, Z. Zhao, C. Chen, Z. Zeng and X. Li, “Two Eyes Are Better Than One: Exploiting Binocular Correlation for Diabetic Retinopathy Severity Grading,” pp. 1-4, 2021.
- [35] W. Zhu, P. Qiu, N. Lepore, O. M. Dumitrascu and Y. Wang, “NNMOBILE-NET: RETHINKING CNN DESIGN FOR DEEP LEARNING-BASED RETINOPATHY RESEARCH,” *School of Computing and Augmented Intelligence*, pp. 1-8, 2023.
- [36] R. Sark, K. Ahmed, H. Wang and Y. Zhang, “Automatic Detection of Diabetic Eye Disease Through Deep Learning Using Fundus Images: A Survey,” *IEEEAccess*, pp. 1-17, 2020.

Appendix

Graduates Attribute Evidence Table

Table 10 below shows how the Graduate's Attributes Evidence was obtained in this project.

Table 10: GAs Table Evidence

Graduates Attribute	Evidence of Graduates' Attribute
GA1	This GA was achieved by defining the complex design engineering problem that requires the engineering solution (section 3.2) A feasible solution is proposed in (section 7) and block diagrams and flow charts (sections 7.2, 12, 12.4 to 12.7.6), were used to solve the problem. Design specifications are provided and explained in detail (section 8) based on the problem identified. A feasibility study was conducted to asses feasibility of the system (section 9). The datasets used in training and evaluation of the deep learning model are analysed and described in detail, Section 12.2.1. The identified CEP is solved through design and implementation of the deep learning model and web application, Section 12.4 and 12.7, by applying fundamental engineering and scientific concepts. Research was done to understand traditional DR diagnosis methods and current advancements to counteract the limitations brought by these methods, Section 5.
GA2	Scientific, mathematical and engineering concepts have been applied to solve complex engineering design problem identified. Mathematical concepts behind the working principle of convolutional neural networks have been applied to design and implement the deep learning model proposed. The concepts learned from Artificial Intelligence module (ENEL4AI) were applied to develop and evaluate the deep learning algorithm proposed (Section 12.4.1.3.2 – 12.7.6). The knowledge acquired from software engineering modules (ENEL2SE and ENEL3SF) was applied to develop the fronted and the backend of the system (Section 12.6 – 12.7). The knowledge concerning the diabetic retinopathy diagnosis was acquired through the conducted research and applied to understand and solve the problem identified.
GA3	Design objectives are outlined in Section 4 and constraints in Section 6. The system block diagrams and flow charts were used to solve the complex engineering problem (sections 7.2, 12, 12.4- 12.7). Web application and the deep learning model were designed before implementation. Model training, evaluation and testing was done using relevant evaluation metrics in Section 12.4.1.3. The functioning of the backend (uvicorn server) of the system was tested and found to be functional as expected, Section 12.7.6. The system was then integrated with the trained model and tested. The system worked as expected, Section 12.7.5.2 and appendix.

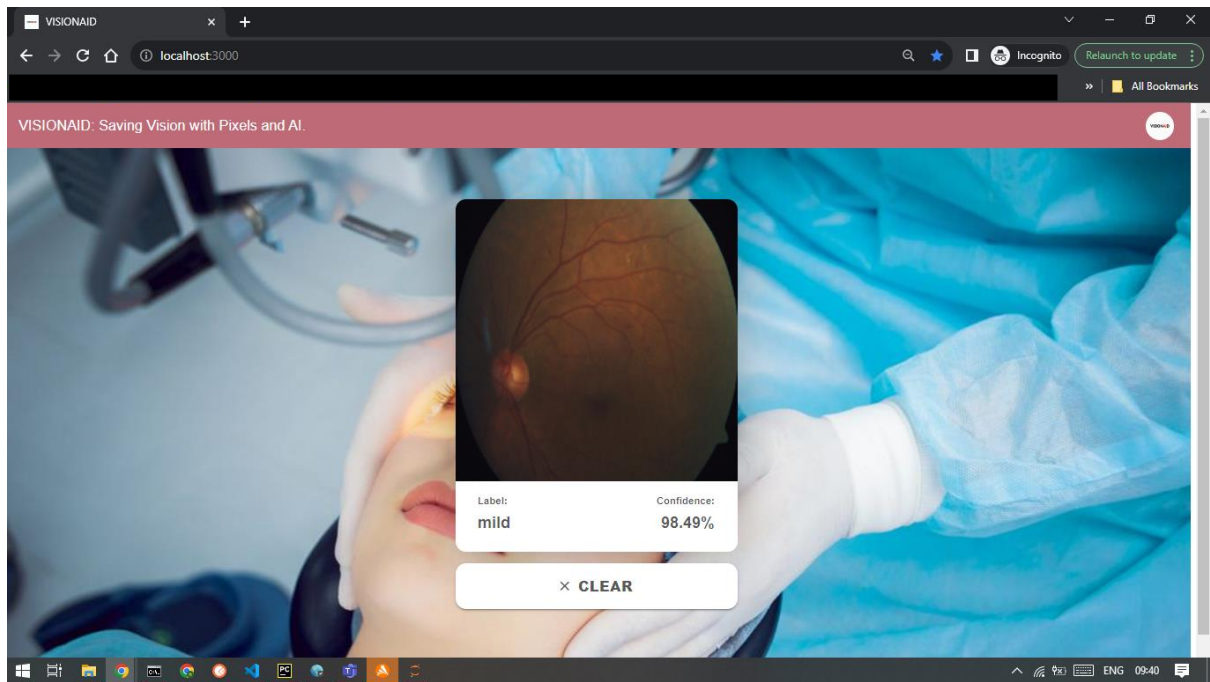


Figure 42: Mild DR Diagnostic Results

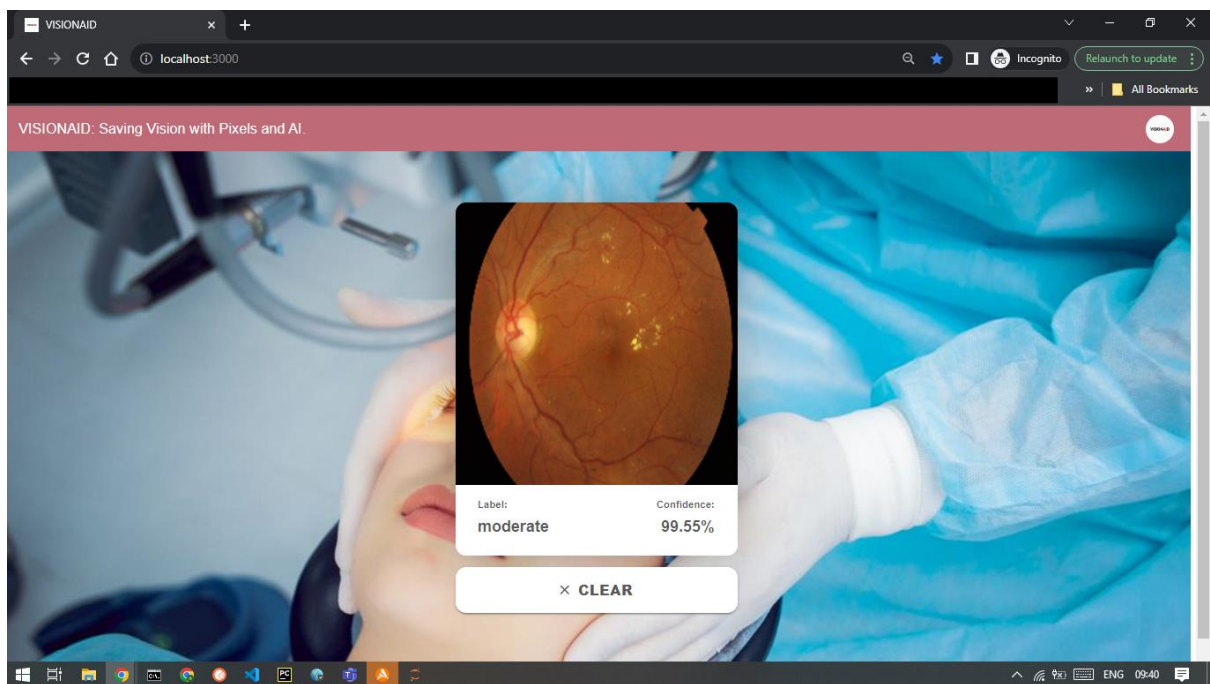


Figure 43: Moderate DR Diagnostic Results

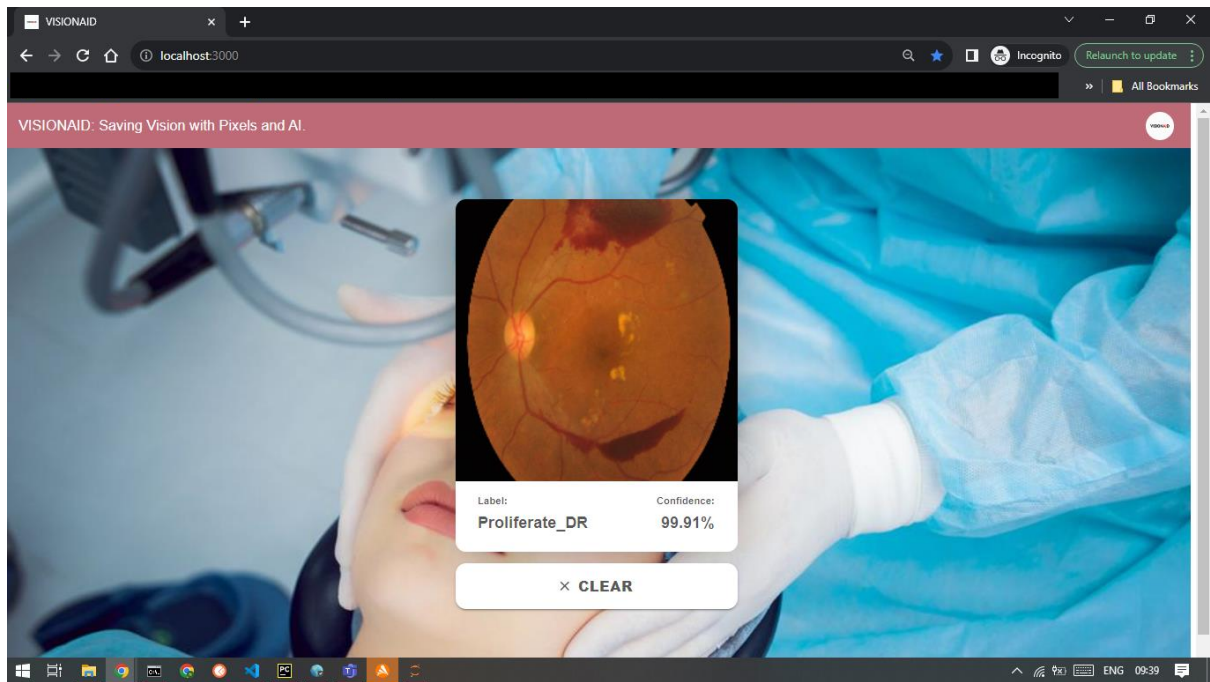


Figure 44: Proliferate DR Diagnostic Results

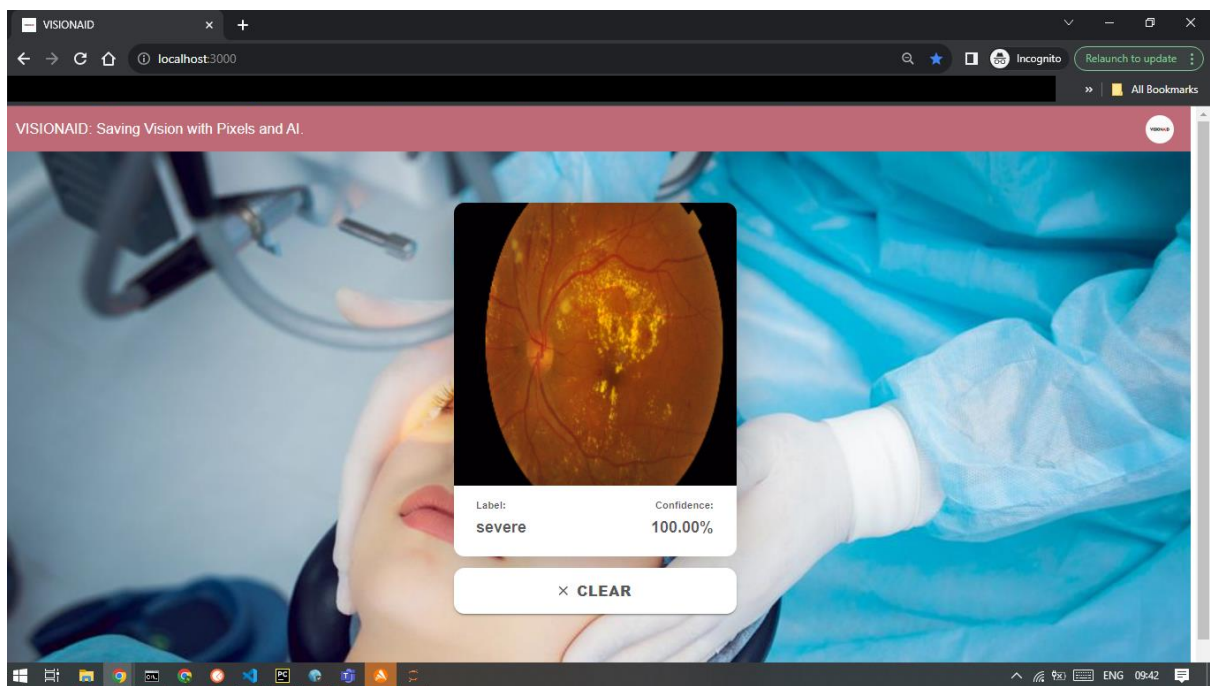


Figure 45: Severe DR Diagnostic Results