DIABETIC RETINOPATHY PREDICTION USING XCEPTION (DEEP LEARNING APPROACH)

A PROJECT REPORT PHASE – I

Submitted by

SANJAY H	142221104110
VIGNESHWAR S	142221104156
VISHWA M	142221104161

In Partial Fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SRM VALLIAMMAI ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report **" DIABETIC RETINOPATHY PREDICTION USING XCEPTION (DEEP LEARNING APPROACH)**" is the Bonafide work of "**Sanjay H (142221104110), Vigneshwar S (142221104156) and Vishwa M(142221104161)**" who carried out the work under my supervision during the academic year 2024-2025.

SIGNATURE	**SIGNATURE**
Dr. B. VANATHI,M.E.,Ph.D.,	**Dr. C.PABITHA, ME., Ph.D**
Professor & Head	**Associate Professor (Sr.G)**
Department of CSE,	Department of CSE.
SRM Valliammai Engineering College,	SRM Valliammai Engineering College,
Kattankulathur-603 203.	Kattankulathur-603 203.

Submitted for the university examination held on _________________
at SRM Valliammai Engineering College, Kattankulathur.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely express our gratitude in depth to our esteemed Founder Chairman & Chancellor **Dr. T. R. Paarivendhar, Thiru. Ravi Pachamootho** Chairman,**Mrs. Padmapriya Ravi** Vice Chairman, **Ms. R. Harini** Correspondent, SRM VALLIAMMAI ENGINEERING COLLEGE for the patronage on our welfare rooted in the academic year.

We express our sincere gratitude to our respected Director **Dr. B. Chidhambararajan, M.E., Ph.D.,** for his constant encouragement, which has been our motivation to strive towards excellence.

We thank our sincere Principal **Dr. M. Murugan, M.E., Ph.D.,** for his constant encouragement, which has been our motivation to strive towards excellence.

We extend our hand of thanks to our Head of the Department**, Dr. B. Vanathi, M.E., Ph.D.,** Professor for her unstinted support.

We are grateful to our internal **guide Dr. C.Pabitha , M.E., Ph.D.,**.Associatet Professor without whose invaluable guidance and encouragement,this project would not have been a success.

We also like to thank all Teaching and non-teaching staff members of our department, for their support during the project. We finally thank our friends and family for their support during the project.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PG NO
	ABSTRACT -ENGLISH	v
	ABSTRACT – TAMIL	vi
	LIST OF FIGURES	viii
1	**INTRODUCTION**	2
	1.1 PROBLEM OVERVIEW	2
	1.2 PROBLEM STATEMENT	3
	1.3 OBJECTIVE	4
2	**LITERATURE SURVEY**	6
	2.1 LITERATURE REVIEW	6
	2.2 EXISTING SYSTEM	12
	2.3 PROPOSED SYSTEM	13
3	**SOFTWARE REQUIREMENTS AND SPECIFICATION**	16
	3.1. SOFTWARE REQUIREMENT	16
	3.2. PYTHON AND VSCODE	16
	3.2.1. ABOUT THE PYTHON SHELL AND IDLE	16
	3.2.2. RUNNING PYTHON LOCALLY	18
	3.2.3. SYNTAX AND SEMANTICS	19
	3.2.4. INDENTATION	19
4	**IMPLEMENTATION OF DR DETECTION METHOD**	21
	4.1. BASIC SYSTEM LEVEL BLOCK DIAGRAM	21
	4.2. PRE-PROCESSING STAGE (PPS)	22
	4.3. FEATURE EXTRACTION	26
	4.4. FLOWCHART	28
	4.5. ALGORITHM	29
5.	**XCEPTION MODEL**	32
	5.1. DATA ACQUISITION	32
	5.2. DATA PREPROCESSING	32
	5.3. MODEL SELECTION AND ARCHITECTURE (XCEPTION):	34
	5.4. MODEL TRAINING DESCRIPTION:	34
	5.5. MODEL EVALUATION DESCRIPTION:	35
	5.6. XCEPTION ARCHITECTURE OVERVIEW	36
6.	**SAMPLE CODE**	41
7.	**SYSTEM TESTING**	45
	7.1 UNIT TESTING	45
	7.2 INTEGRATION TESTING	45
	7.3 FUNCTIONAL TEST	46
	7.4 SYSTEM TEST	46
	7.5 TEST STRATEGY AND APPROACH	47
8.	**RESULT AND CONCLUSION**	50
9.	**REFERENCE**	54

ABSTRACT

Diabetic retinopathy (DR) is a severe eye condition caused by prolonged high blood sugar levels, which damages the blood vessels in the retina and can lead to vision loss or blindness if left untreated. Early detection and intervention are crucial for preventing vision impairment. This project aims to develop a deep learning-based system using the Xception architecture for the automated detection and classification of DR from retinal fundus images. By training on a dataset of labelled retinal images, the model learns to identify and assess the severity of DR, offering a non-invasive, cost-effective, and rapid screening tool. This tool has the potential to aid ophthalmologists in the early diagnosis of DR, thereby facilitating timely treatment and effective disease management. The project demonstrates the efficacy of deep learning in improving diagnostic accuracy and efficiency in ophthalmology, particularly in underserved regions where access to specialized care may be limited. The automated detection model developed in this project offers significant potential as a rapid, cost-effective screening tool that can facilitate early diagnosis and improve patient outcomes. Ultimately, this project underscores the role of deep learning in transforming ophthalmic diagnostics and advancing preventive healthcare for diabetic patients.

சுருக்கம்

நீரிழிவு ரெட்டினோபதி (டிஆர்) என்பது நீண்ட காலமாக உயர் இரத்த சர்க்கரை அளவுகளால் ஏற்படும் ஒரு கடுமையான கண் நிலை ஆகும், இது விழித்திரையில் உள்ள இரத்த நாளங்களை சேதப்படுத்தும் மற்றும் சிகிச்சையளிக்கப்படாவிட்டால் பார்வை இழப்பு அல்லது குருட்டுத்தன்மைக்கு வழிவகுக்கும். பார்வைக் குறைபாட்டைத் தடுப்பதற்கு முன்கூட்டிய கண்டறிதல் மற்றும் தலையீடு முக்கியமானது. இந்தத் திட்டமானது, விழித்திரை ஃபண்டஸ் படங்களிலிருந்து DRஐ தானியங்கு கண்டறிதல் மற்றும் வகைப்படுத்துதலுக்கான Xception கட்டமைப்பைப் பயன்படுத்தி ஆழ்ந்த கற்றல் அடிப்படையிலான அமைப்பை உருவாக்குவதை நோக்கமாகக் கொண்டுள்ளது. லேபிளிடப்பட்ட விழித்திரை படங்களின் தரவுத்தொகுப்பில் பயிற்சியளிப்பதன் மூலம், DR இன் தீவிரத்தை அடையாளம் காணவும் மதிப்பிடவும் மாடல் கற்றுக்கொள்கிறது, ஆக்கிரமிப்பு இல்லாத, செலவு குறைந்த மற்றும் விரைவான திரையிடல் கருவியை வழங்குகிறது. இந்த கருவி கண் மருத்துவர்களுக்கு DR இன் ஆரம்பகால நோயறிதலுக்கு உதவும் திறனைக் கொண்டுள்ளது,

இதன் மூலம் சரியான நேரத்தில் சிகிச்சை மற்றும் பயனுள்ள நோய் மேலாண்மைக்கு உதவுகிறது. கண் மருத்துவத்தில் கண்டறியும் துல்லியம் மற்றும் செயல்திறனை மேம்படுத்துவதில் ஆழமான கற்றலின் செயல்திறனை இந்த திட்டம் நிரூபிக்கிறது, குறிப்பாக சிறப்பு கவனிப்புக்கான அணுகல் குறைவாக இருக்கும் குறைவான பகுதிகளில். இந்தத் திட்டத்தில் உருவாக்கப்பட்ட தானியங்கு கண்டறிதல் மாதிரியானது விரைவான, செலவு குறைந்த ஸ்கிரீனிங் கருவியாக குறிப்பிடத்தக்க ஆற்றலை வழங்குகிறது, இது ஆரம்பகால நோயறிதலை எளிதாக்கும் மற்றும் நோயாளியின் விளைவுகளை மேம்படுத்தும். இறுதியில், இந்த திட்டம், நீரிழிவு நோயாளிகளுக்கு கண் நோய் கண்டறிதல் மற்றும் தடுப்பு சுகாதாரத்தை மேம்படுத்துவதில் ஆழ்ந்த கற்றலின் பங்கை அடிக்கோடிட்டுக் காட்டுகிறது.

LIST OF FIGURES

FIGURE NO	DESCRIPTION	PAGE NO.
1.	STAGES OF DIABETIC RETINOPATHY	3
2.	BLOCK DIAGRAM	21
3.	Pre-Processing Stage (PPS)	22
4.	Colour Fundus Image	23
5.	Cropped image	24
6.	Grayscale image	24
7.	Binary image	25
8.	Flowchart	28
9.	Training Model of 20 epoch	50
10.	95.66% accuracy of final epoch	50
11.	Direct Output	50
12.	Web Output	51
13.	Web Output	51

CHAPTER 1

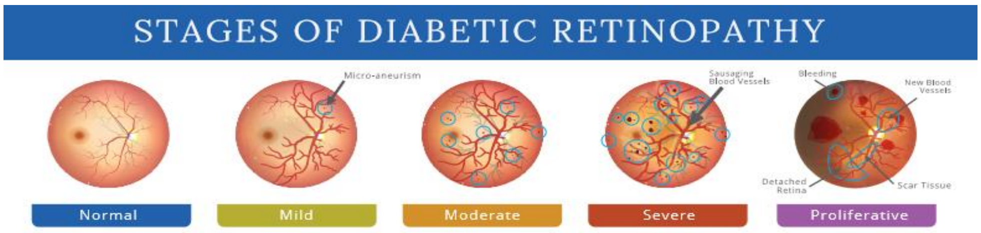
1. INTRODUCTION

1.1 PROBLEM OVERVIEW

Diabetes or diabetes mellitus is a metabolic disease in which the person body produces an inadequate amount of insulin to produce high blood sugar. In India itself, more than 62 million people are suffering from diabetes. The people who are suffering from diabetes for more than 20 years has 80% chance of causing diabetic retinopathy According to the International Diabetes Federation, the number of adults with the diabetes in the world is estimated to be 366 million in 2011 and by 2030 this would have risen to 552 million. The number of people with type 2 diabetes is increasing in every country 80% of people with diabetes live in low-and middle-income countries. India stands first with 195% (18 million in 1995 to 54 million in 2025). Previously, diabetes mellitus (DM) was present, largely, among the urban population in India. Recent studies clearly show an increasing prevalence in rural areas as well. Indian studies show a 3-fold increase in the presence of diabetes among the rural population over the last decade or so (2.2% in 1989 to 6.3% in 2003) In India, Study shows the estimated prevalence of type 2 diabetes mellitus and diabetic retinopathy in a rural population of south India are nearly 1 of 10 individuals in rural south India, above the age of 40 years. Diabetic retinopathy is a state of eye infirmity in which damage arises due to diabetes mellitus. It is one of the prominent reasons behind blindness. The increased blood sugar due to diabetes incorporated damage to the tiny blood vessels in the retina thereby causing diabetic retinopathy At least 90% of new cases could be reduced with proper medication as well as frequent monitoring of the eyes It primarily affects the retinas of both the eyes, which can lead to vision loss if it is not treated. Poorly controlled blood sugars, high blood pressure, and high cholesterol increase the risk of developing Diabetic retinopathy. The earlier work in the detection of varies stages DR based on explicit feature extraction & classification by using various Image Processing techniques & Machine learning algorithm respectively. Though high accuracy can be achieved using these methods but diagnosing diabetic retinopathy based on the explicit extraction of features is an intricate procedure. Due to development of Computer vision in recent times & availability of large dataset, it is now possible to use a deep Neural network for the detection & classification of Diabetic retinopathy. Hence, several methods have been proposed based on the deep neural network for the classification of Diabetic retinopathy based on severity A major difficulty of fundus image classification using the deep neural network is high variability, especially in the case of retinal proliferation and retinal detachment of new blood vessels, which lowers the accuracy of the network. The method proposed in this paper aims at detecting the various stages of Diabetic Retinopathy by using U-Net segmentation with region merging & Convolutional Neural Network. The retinal segmentation is the process of automatic detection of boundaries of blood vessels within the retina. This allows classifier to learn important features such as retinal proliferation and retinal detachment. The data lost during retinal segmentation is retracted through region merging.

1.2 PROBLEM STATEMENT

Diabetic retinopathy (DR) is a leading cause of blindness among adults worldwide, primarily affecting individuals with diabetes. The condition often progresses without noticeable symptoms until significant damage has occurred, making early detection vital to preventing irreversible vision loss. This project aims to develop a
Deep learning-based model capable of analyzing retinal fundus images to predict the presence and severity of diabetic retinopathy.



1.STAGES OF DIABETIC RETINOPATHY

1.3 OBJECTIVE

The objective of this project is to develop and implement a deep learning-based model for the automated detection and classification of diabetic retinopathy from retinal fundus images.This model aims to assist in early diagnosis by identifying the presence and severity of diabetic retinopathy, providing a non-invasive, cost-effective, and rapid screening tool.The goal is to enhance the diagnostic accuracy above 92% and efficiency in ophthalmology, supporting timely treatment and reducing the risk of vision loss in patients with diabetic retinopathy. As the number of diabetes affected people is increasing worldwide, the need for automated detection methods of diabetic retinopathy will increase as well. To automatically detect diabetic retinopathy, a computer has to interpret and analyze digital images of the retina. Automatic detection of Diabetic Retinopathy (ADDR) is a fully automated system for detection of Diabetic Retinopathy.

CHAPTER 2

LITERATURE SURVEY:

2.1 LITERATURE REVIEW:

2.1.1 Diabetic Retinopathy Prediction based on CNN and AlexNet model

Diabetic retinopathy is highlighted as a major cause of blindness among diabetic patients, with a growing prevalence worldwide. The traditional diagnostic process for DR involves manual inspection of retinal fundus images by ophthalmologists, which is time-consuming, costly, and prone to inaccuracies. The paper advocates for an AI-driven solution using CNNs to streamline and improve the DR detection process. The study aims to develop a CNN-based model to detect and classify DR severity using the APTOS dataset, a large, publicly available set of retinal fundus images labelled by DR stage. The literature review covers various machine learning techniques and CNN architectures previously used for DR detection, including popular datasets like Kaggle's Diabetic Retinopathy Detection Dataset, EyePACS, and Messidor-2. Various methods are discussed, including feature extraction, segmentation techniques, and ensemble models, highlighting CNN's superior performance.

Author Name: Ritesh Chandra, Sadhana Tiwari, Shashi Shekhar Kumar, Sonali Agarwal.

Drawbacks:

1. Limited Model Choice: The study only compares a custom CNN an AlexNet, not newer models that might improve accuracy.

2.Generalization Issues: The model was only tested on the APTOS dataset, so it may not work as well on other datasets or real-world images.

3.Lack of Explainability: The model does not explain its decisions, which can make it harder for doctors to trust its predictions.

4.Risk of Overfitting: Without regularization techniques, the model might perform well on training data but poorly on new, unseen data.

2.1.2. Prediction of Diabetic Retinopathy Based on Risk Factors using Machine Learning Algorithms.

The paper discusses the increasing prevalence of diabetes and diabetic retinopathy, particularly in Malaysia, where only a small percentage of diabetic patients undergo regular eye exams. DR often develops without symptoms in its early stages, leading many patients to seek medical help only when the disease has progressed. Current DR screening methods, such as fundus photography, are effective but require specialized skills and equipment, which can be a barrier in resource-limited settings. The data for this study was collected from clinical records at Universiti Teknologi MARA's Ophthalmology Clinic, covering diabetic patients' demographics and health information. The dataset includes features like age, blood pressure, diabetes duration, and complications (e.g., nephropathy, neuropathy). The authors employed three machine learning algorithms: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Logistic Regression (LR), each optimized with various hyperparameters.

Author Name: Nor Tasha Nadira Nor Azamen, Azliza Mohd Ali, Nor Azimah Abd Aziz.

Drawbacks:

1. Limited Dataset: The research was conducted using a relatively small dataset obtained from the records of Type 2 diabetes patients at UiTM Sg Buloh Medical Centre between 2016 and 2019. A larger dataset would improve the reliability and generalization of the models.

2. Retrospective Data Limitation: The dataset was collected retrospectively, relying on existing patient records. This approach can be affected by the accuracy and completeness of documentation by attending physicians, which may lead to missing or inconsistent data.

3. Potential for Missing Data: The study mentions that future improvements should include structured questionnaires to obtain more consistent and complete information on risk factors, avoiding missing data issues.

4. Model Performance Scope: Although logistic regression performed best with an accuracy of 83.78%, other models like SVM and KNN yielded lower accuracies (70.27% and 76.71% respectively). This indicates potential limitations in their predictive power for this specific problem.

5. Model Enhancement Recommendations: The authors suggest adding more diverse datasets and relevant risk factors to enhance the training, testing, and overall performance of the machine learning models for better generalization.

2.1.3. Classification of Diabetic Retinopathy Using Image Pre-processing Techniques

Diabetic retinopathy is a serious eye condition resulting from diabetes and can lead to vision impairment or blindness. Early detection of DR is crucial to prevent severe outcomes. The paper outlines the classification of DR images using pre-processing techniques to enhance the accuracy of deep learning models. Pre-processing techniques, including erosion and histogram equalization, were applied to retinal fundus images from a Kaggle dataset. The CNN (Convolutional Neural Network) model used in the study included one convolutional layer, one max-pooling layer, a flatten layer, and three dense layers, employing the ReLU and SoftMax activation functions. The Adam optimizer was utilized for training.

Author Name: Kavya Duvvuri, Harshitha Kanisettypalli, Masabattula Teja Nikhil, Suja Palaniswamy.

Drawbacks:

Dataset Limitation: The study used a dataset with specific distributions of DR stages, which might limit the generalizability of the model. A larger, more diverse dataset could improve the robustness and applicability of the results.

Single Pre-processing Focus: The research primarily focused on only a few pre-processing techniques, such as erosion and histogram equalization. While this improved accuracy, the document suggests that additional or alternative techniques might further enhance the model's performance.

Model Complexity: The CNN model implemented was relatively simple, comprising one convolutional layer, one max-pooling layer, and a few dense layers. More complex architectures like deeper or more sophisticated CNNs (e.g., ResNet or Inception) might yield higher accuracy but were only partially evaluated.

Explainability and Validation: Although the Grad-CAM explainable AI technique was used, further validation through clinical expert assessment would strengthen the claims about model effectiveness and highlight regions contributing to the predictions.

Comparison with More Advanced Models: While the paper included comparisons with Alex Net, VGG-16, and ResNet-50, the overall performance of these models without significant tuning may not reflect their potential. A more detailed comparison, including model fine-tuning or ensemble approaches, could provide better insights.

2.1.4. **Detection and Classification of Diabetic Retinopathy using Pretrained Deep Neural Networks.**

Diabetic retinopathy is a severe eye condition affecting people with diabetes, potentially leading to vision impairment or blindness. Early detection and classification are crucial for effective management. Current manual methods of diagnosis are time-consuming and can vary between specialists. Automated detection using deep learning offers a promising solution. The research uses pretrained deep convolutional neural networks, specifically VGG-16 and MobileNetV2, to classify fundus images into five DR stages: no DR, mild, moderate, severe, and proliferative DR .The APTOS 2019 dataset from Kaggle, consisting of 3662 retinal fundus images, was used for training and testing, resized to 224x224 pixels for compatibility with the networks. The models were fine-tuned to adapt to the specific DR classification task, employing techniques such as feature extraction and adding a fully connected layer for final classification.

Author Name: Abini M.A ,S. Sridevi Sathya Priya

Drawbacks:

Dataset Quality and Variability: The APTOS 2019 dataset, while comprehensive, includes images with varying quality, exposure, and focus. These inconsistencies can affect the robustness of the model, as it may not perform as well on images from different sources or lower-quality real-world data.

Model Generalizability: The study primarily evaluates models on a single dataset. This limits the understanding of how well the models generalize to other datasets or diverse populations. Testing on additional datasets or cross-validation with multiple sources would provide better insights into model performance across different environments.

Complexity of Classification: The paper highlights that distinguishing between normal and early stages of DR (e.g., mild vs. no DR) is challenging. The results indicate that while MobileNetV2 performs well, the confusion matrices show some difficulties in correctly classifying less severe stages.

Interpretability of Model Decisions: While pretrained CNNs provide high accuracy, their decisions are often considered "black boxes." The document does not address the use of explainability techniques (e.g., Grad-CAM or LIME) .

2.1.5. Detection of Diabetic Retinopathy with Retinal Images using CNN

Diabetic retinopathy is a vision-threatening condition for diabetics. The document states that manual diagnosis by ophthalmologists is time-consuming, prompting the need for an automated, efficient solution. The methodology includes data acquisition, pre-processing (data augmentation techniques such as rotation and shearing), model training, and classification of retinal images. The CNN model architecture includes convolutional, pooling, and dense layers, optimized to extract features and make classifications. The results are favorable, showing that this CNN-based approach surpasses traditional methods in accuracy. The paper also includes performance metrics and a comparison of the proposed approach with other models, demonstrating its effectiveness for clinical applications.

Author Name: Samiya Majid Baba, Indu Bala

Drawbacks:

Data Size and Generalization: The dataset consists of only 757 retinal images for training and 151 images for testing. While this produced high accuracy within this dataset, the model may struggle with generalization when applied to more diverse or larger datasets from different sources. Expanding the dataset size and diversity could improve robustness.

Class Imbalance and Data Augmentation: Though data augmentation techniques were used to mitigate class imbalance, these methods alone may not fully address the issue. The model’s performance could be affected if certain stages of diabetic retinopathy are underrepresented, potentially impacting sensitivity and specificity for more nuanced cases.

Model Complexity and Computational Requirements: CNNs, particularly with deeper architectures, require significant computational power for training and real-time classification. This could be a limitation in settings with limited computational resources, particularly in lower-resource healthcare facilities.

Limited Real-World Validation: The model was validated on a relatively small testing set, and no external validation on an independent dataset is reported. Real-world performance might differ due to variations in patient demographics, imaging protocols, and equipment, which could impact reliability across diverse clinical settings.

2.2 EXISTING SYSTEM

The existing system for diabetic retinopathy detection predominantly relies on manual examination by trained ophthalmologists, who analyse retinal fundus images to diagnose the condition. Additionally, the manual process is susceptible to human error, which can lead to inconsistent results. Recent developments have focused on automating the detection of diabetic retinopathy using deep learning techniques with accuracy over 92%.These existing models can extract features from retinal images, significantly improving diagnostic accuracy and consistency using CNN.

Traditional Machine Learning Systems Approach: Early systems for diabetic retinopathy detection typically used traditional machine learning techniques such as Support Vector Machines (SVMs), k-nearest neighbors (KNN), and decision trees. These methods often relied on handcrafted features extracted from retinal images, such as blood vessel detection, exudate detection, or texture analysis.

Drawbacks:

Feature Engineering: Handcrafted features can be insufficient for capturing complex patterns in retinal images.

Low Accuracy: These systems tend to have lower accuracy compared to deep learning methods due to the reliance on predefined feature extraction methods.

Limited Scalability: Not ideal for handling large, diverse datasets and new data without significant re-engineering.

Convolutional Neural Networks (CNNs) Approach: CNN-based systems became widely adopted for diabetic retinopathy detection, where CNNs automatically learn to extract features from images without manual feature engineering. ResNet, InceptionNet, and VGG are commonly used architectures. These systems are usually applied for classification (e.g., determining the severity of DR from images).

Drawbacks:

High Data Requirements: CNNs require a large amount of labelled data for training, which can be a challenge for medical datasets where annotations are expensive to obtain.

Black Box Nature: Limited interpretability, which is a critical concern in medical domains where understanding the "why" behind decisions is important.

2.3 PROPOSED SYSTEM

The proposed system aims to use a Xception model a deep learning technique to automatically detect diabetic retinopathy from retinal fundus images. The system will be trained using a labelled dataset of eye images, allowing it to learn and identify features that indicate the presence and severity of diabetic retinopathy. Preprocessing steps, such as image resizing, normalization, will be applied to improve image quality and model performance. Here, we use Xception, which is more efficient due to its transfer learning making it faster and more effective for classification than other methods. This approach offers a fast, non-invasive, and cost-effective tool for detecting diabetic retinopathy, especially in resource-limited settings.

Advantages of the System:

Data Preprocessing: Before feeding images into the model, preprocessing steps like image resizing and normalization are applied. These steps ensure that the images are uniform in size, making it easier for the model to process them consistently. Normalization adjusts the image pixel values, which enhances image quality and improves the model's ability to detect subtle features in the retina, ultimately boosting performance.

Using the Xception Model: The Xception model is chosen due to its efficiency and strong performance in image classification tasks. Xception is an advanced convolutional neural network (CNN) that employs depth wise separable convolutions. This unique structure allows it to extract meaningful features from images while keeping computational costs lower than traditional CNNs. Additionally, transfer learning is applied, where the model leverages knowledge from prior training on similar image datasets, making it faster to train and highly effective for classification in this specific medical context.

Benefits for Diabetic Retinopathy Detection:

Speed and Accessibility: This automated approach offers a quick and non-invasive method for detecting diabetic retinopathy, which is crucial in busy clinical environments.

Cost-Effectiveness: By automating the detection process, this system reduces the reliance on specialized medical professionals for initial screenings, making it an affordable tool for early diagnosis.

CHAPTER 3

3.SOFTWARE REQUIREMENTS AND SPECIFICATION

3.1. SOFTWARE REQUIREMENT

• Operating Systems -Windows

• Programming Languages - Python

• Tools Used -VSCode

• Front End -HTML,CSS

• Processor -Intel Pentium Dual Core i5 above

• Speed - 2.2GHz

• RAM -8GB

• Hard Disk - 200 GB

3.2. PYTHON AND VSCODE

3.2.1. About the Python Shell and idle

Python is an interpreted programming language, which means that code can be executed line by line without needing to go through a compilation step, as required in languages like C or FORTRAN. This makes Python particularly convenient for quick scripting and testing, as changes to the code can be tested immediately. Starting the Python interpreter is as simple as typing `python` into a command line, where you can begin entering commands directly. For a more integrated experience, developers often use a code editor like Visual Studio Code (VSCode), which is highly regarded for Python development due to its extensive features tailored to Python workflows. VSCode allows you to open multiple editor windows, manage multiple files, and save your work easily with the Save or Save As options from the File menu. Once you've saved a script, such as `MyScript.py`, you can run it from within VSCode by selecting Run from the editor's toolbar, or directly from the terminal by typing `python MyScript.py`.

VSCode's Python extension provides a few features that enhance the coding experience. It offers syntax highlighting, which colorizes keywords and functions, making the code more readable. Additionally, it includes auto-indentation to follow Python's strict indentation rules, as well as code completion, with popup suggestions for functions, arguments, and libraries that help speed up coding and reduce errors. These features make it easy to write, edit, and debug Python code directly in the editor. Moreover, VSCode's support for integrated terminal and command palette further streamlines the workflow, allowing developers to execute code without leaving the editor.

For practice, users can create a small script, such as a note or a simple calculation (`print("Hello, world!")`), save it, and run it to see the output. They can also reopen the file in new editor windows to get familiar with handling multiple files in VSCode. This workflow can be extended to work with larger Python projects by organizing code into functions and storing them in separate files, making code modular and reusable. While VSCode is a versatile and powerful tool, Python also comes with an integrated environment known as IDLE. IDLE provides a simpler interface that includes a Python interpreter shell and an editor for writing scripts. To start IDLE, you can type `idle` in the command line, which opens a Python Shell window where you can test code directly and create or open new Python files. The IDLE editor includes basic debugging tools and can run scripts directly from within the editor window, making it an alternative to VSCode for smaller or single-file projects.

Whether working in VSCode or IDLE, you can easily exit the Python interpreter by typing `Ctrl+D` in the command line. Both environments allow developers to store functions and reusable code in files that can be loaded later, saving time and preventing the need to retype code. In VSCode, this is especially powerful as it allows you to handle complex Python projects that may involve multiple files and dependencies. As you develop in Python, both VSCode and IDLE offer different benefits; VSCode is ideal for complex projects with its extensive tooling support, while IDLE provides a straightforward, accessible setup for simpler tasks or for quickly testing code. Both options make Python development efficient, allowing for flexibility in testing, debugging, and managing scripts.

3.2.2. Running Python Locally

When using Visual Studio Code (VSCode) for Python development, it's important to note that many Python-based exercises can be done locally on your machine without needing additional data files or specialized extensions. If you have a personal computer, you can download Python from [python.org](https://www.python.org/), which offers installers for Mac, Linux, and Windows. For example, MacPython provides an integrated Python environment that includes a full interpreter and editor setup for both OS9 and OSX, and is often preferred over IDLE by Mac users.

VSCode can serve as your primary development environment for Python. If your course provides Python scripts in plain text (like `phys.py`), these can be downloaded and placed in your Python project directory in VSCode, allowing you to work on assignments across different platforms without compatibility issues. However, be aware that some compiled Python extension modules (e.g., `veclib.so`) may require specific versions of Python or specific hardware, which might not be compatible across all operating systems.

Throughout your assignments, when instructions indicate "start up the Python interpreter," you have options: you can choose between using the simple command-line interpreter, IDLE, or an IDE like VSCode. For tasks that require graphical outputs, make sure you're connected to a display-enabled environment (like an xterm if you're working remotely) so that graphics can be rendered properly on your screen. VSCode is highly flexible in this respect, as it supports a range of display libraries and rendering for data visualization outputs (e.g., matplotlib graphs).

If you're working with basic Python programs and standard modules, any personal computer should suffice. In VSCode, you can write scripts in the editor and execute them directly, either by running them in the integrated terminal or through the Run Python File command. Exercises involving file I/O, such as reading and writing text files, can be done on your local machine as well—just ensure any necessary data files are downloaded to your system in advance. VSCode offers a versatile setup that allows for consistent Python development across platforms and supports most Python-based coursework. Whether you're running basic scripts or working with data files, VSCode, alongside standard Python installations, can accommodate a variety of tasks without needing specialized setup or resources beyond what's available on [python.org](https://www.python.org/).

3.2.3. Syntax And Semantics

Python syntax and semantics

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal

3.2.4. Indentation

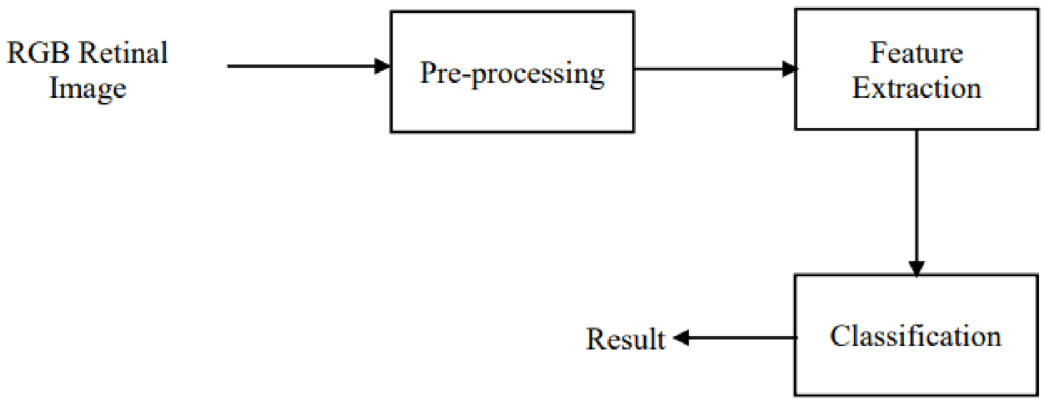
Python syntax and semantics & Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is also sometimes termed the off-side rule. The enforcement of indentation in Python makes the code look neat and clean. This results in Python programs that look similar and consistent.

CHAPTER 4

4. Implementation of DR Detection Method

4.1. Basic system level block diagram



2. BLOCK DIAGRAM

Automatic detection of Diabetic Retinopathy (ADDR) is a fully automated system for detection of Diabetic Retinopathy (DR). Figure shows the block diagram of ADDR. Input to this system is a fundus image which is part of human eye that can be seen through the pupil. Fundus image [1,4] is the interior surface of the eye, opposite the lens, and includes the retina, optic disc, macula, Blood vessels and fovea. As the quality of the image is not satisfactory because of noise, bad contrast, uneven illumination etc. pre-processing is used to get better results.

The methodology is made up of three fundamental parts,

Pre-processing

The aim of pre-processing is to attenuate the noise, to improve the contrast and to correct the non-uniform illumination. In the RGB images, the green channel exhibits the best contrast between the vessels and background while the red and blue ones tend to be more noise. Hence green channel is used for further processing.

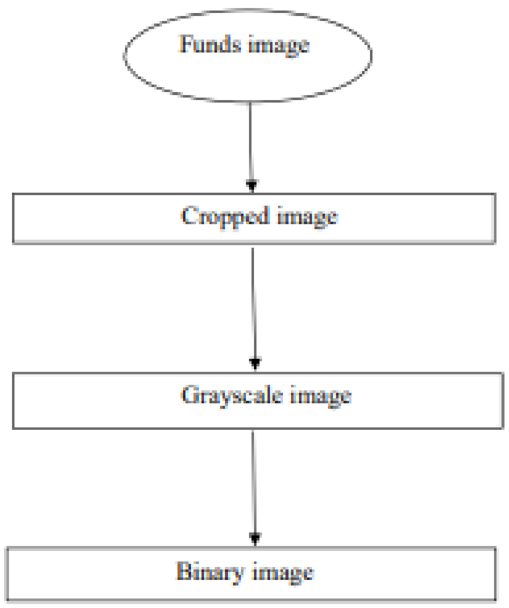
Feature Extraction

Objective of Feature Extraction is to select all Micro aneurysms present in the pre-processed image. Micro aneurysms appear as isolated patterns and are disconnected from the vessels. The features of micro aneurysms can be extracted based on shape, size and intensity level. Micro aneurysms are dark reddish in colour, they appear as small red dots of 10 to 100microns diameter and are circular in shape

Classification

After the detection of Micro aneurysms, classification groups the eye images as either diseased or normal depending on the count of detected micro aneurysm.

4.2. Pre-Processing Stage (PPS)



3. Pre-Processing Stage (PPS)

In detecting abnormalities associated with fundus image, the images must be pre-Processed to correct the problems of uneven illumination problem, nonsufficient contrast between exudates and image background pixels and presence of noise in the input fundus image. Aside from afore mentioned problems, this section is also responsible for color space conversion and image size standardization for the system. This section, which is Pre- Processing stage, can be regarded as the bedrock of this research work. The block diagram of the sub sections that constitute the Pre-Processing stage (PPS)

Colour Fundus Image

The input fundus image is an RGB image. This image is a retinal fundus image of patient. This fundus image can be either normal or defected. We used this image as input image. We must apply some processes to this image to detect the diabetic retinopathy. In our database, there are such 30 fundus images in which 10 images are normal, 10 images are haemorrhage and remaining 10 are exudates.

Syntax

4. Colour Fundus Image

A = imread(filename, fmt)

Description

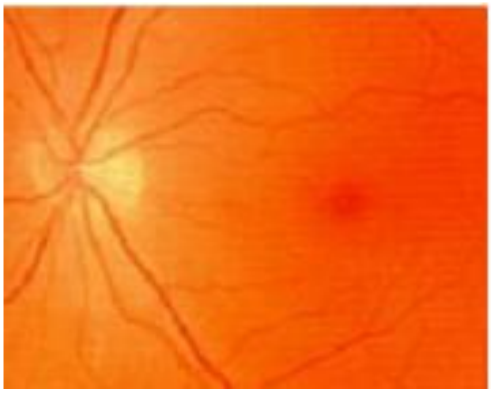
A = imread(filename, fmt) reads a grayscale or color image from the file specified by the string filename. If the file is not in the current directory, or in a directory on the MATLAB path, specify the full pathname. The text string fmt specifies the format of the file by its standard file extension. For example, specify 'gif' for Graphics Interchange Format files. To see a list of supported formats, with their file extensions, use the informants function. If imread cannot find a file named filename, it looks for a file named.

Cropped image

The input image must have to crop at specific size, because the whole part of fundus image is not defected. So, we have to choose only the area near the pupil. Only this area more affects the patients vision ability.

Syntax

I1= imcrop(I, rect)


5. Cropped image

Description

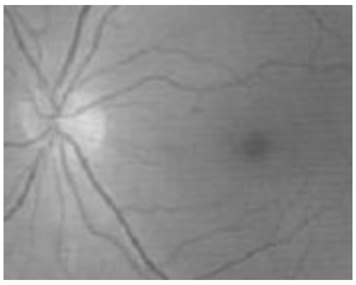
I = imcrop creates an interactive Crop Image tool associated with the image displayed in the current figure, called the target image .I1 = imcrop(I, rect) crops the image I. rect is a four-element position vector[xmin ymin width height] that specifies the size and position of the crop rectangle.

Grayscale image

The Grayscale image exhibits the best contrast between the vessels and background while the red and blue ones tend to be more noise. Hence Grayscale image is used for further processing. the retinal blood vessels appear darker in the grey image.

Syntax

I = rgb2gray(RGB)


6. Grayscale image

Description

I = rgb2gray(RGB) converts the true color image RGB to the grayscale intensity image I. rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

Binary image

For scanning this image, we have to convert image into bitwise binary image. In binary image we provide level on which the image divides in two parts i.e. black part and white part. In which the black part shows the defected part of eye.

7. Binary image

Syntax

I1 = im2bw(I, level)

Description

I1 = im2bw(I, level) converts the grayscale image I to a binary image. The output

image I1 replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black). You specify level in the range [0,1], regardless of the class of the input image. The function gray thresh can be used to compute the level argument automatically. If you do not specify level, im2bw uses the value 0.5.

4.3. Feature extraction

1. Feature Extraction with Xception for Diabetic Retinopathy Stages

The Xception model will serve as the backbone for extracting important features from retinal images to identify different stages of diabetic retinopathy. Each stage has distinctive markers, such as microaneurysms, hemorrhages, and exudates, which become more prevalent and severe as the condition progresses. Using the Xception model's deep layers, we can capture these fine details that differentiate each stage.

2. Stage Classification Based on Specific Retinal Features

Each stage of diabetic retinopathy has unique features, and the model can be trained to recognize these:

Normal: Retinal images with minimal or no abnormalities. The model will learn to classify images as "Normal" if no significant features, such as microaneurysms or exudates, are detected.

Mild: In this stage, early signs of diabetic retinopathy appear, such as microaneurysms, which are small red dots or dark spots on the retina. The Xception model can be trained to recognize these patterns based on their circular shape, size, and reddish color. These isolated patterns are a key marker of mild diabetic retinopathy.

Moderate: As the disease progresses to the moderate stage, additional signs like haemorrhages and exudates start to appear more frequently. Haemorrhages are slightly larger dark spots, while exudates appear as bright spots. The model will identify and count these abnormalities, differentiating moderate retinopathy from the mild stage based on their increased frequency and size.

Severe: In severe retinopathy, the retina shows a higher concentration of abnormalities, including larger haemorrhages and multiple exudates spread throughout the retinal image. The model will assess the density and spread of these features, distinguishing severe cases by the higher presence of these abnormalities across a larger retinal area.

Proliferative: This is the most advanced stage of diabetic retinopathy, where new blood vessels form on the retina due to extensive retinal damage. These new vessels are fragile and prone to leaking, causing significant visual impairment. The model will identify this stage by detecting not only the high density of haemorrhages and exudates but also the characteristic signs of neovascularization (new blood vessel growth).

3. Binary Scanning and Pixel Counting

To refine the classification, the binary image can be scanned row-by-row to quantify abnormalities. Using a counter variable, \`c\`, the algorithm increments the count whenever a black pixel (value = 0) is encountered, representing areas where microaneurysms or haemorrhages are present. Different ranges of \`c\` values can correspond to different stages:

Normal: Low \`c\` values (indicating few or no dark spots)

Mild to Moderate: Gradual increase in \`c\` values

Severe to Proliferative: High \`c\` values due to extensive abnormal areas.

4. Exudate Detection for Severity Analysis

Exudates are detected by analysing the mean and standard deviation of pixel intensity in grayscale images. A matrix is formed based on these values, capturing variations in brightness that can distinguish between stages of severity. The Neural Network Toolbox can assist in this step by using these metrics for further differentiation of exudate-related stages.

5. Classification Output

By combining the feature extraction, pixel counting, and exudate detection steps, the Xception model can classify retinal images into the following five stages:

Normal: Minimal abnormalities.

Mild: Few microaneurysms present.

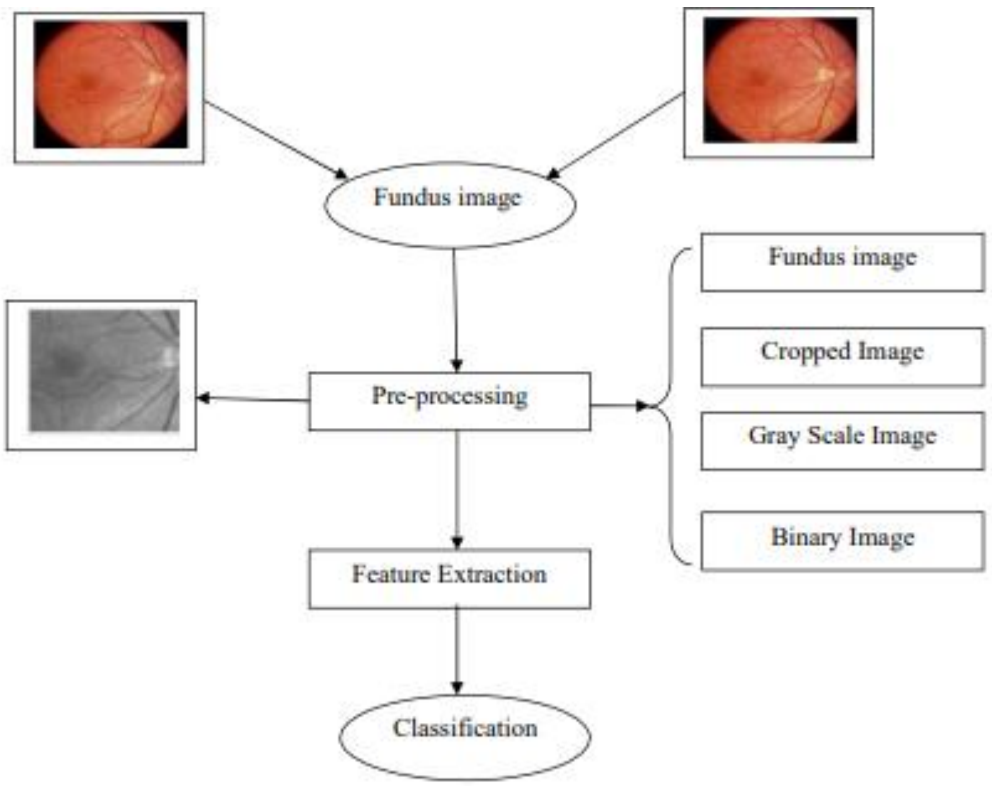
Moderate: Increased microaneurysms and some exudates.

Severe: High presence of haemorrhages and exudates across a larger area.

Proliferative: Signs of neovascularization and extensive abnormalities.

This classification system, powered by the Xception model, provides an efficient, accurate, and scalable method to assess and classify diabetic retinopathy stages, offering valuable insights for early detection and treatment planning.

4.4. Flowchart



8. Flowchart

4.5. Algorithm

1. Initialization

Step 1: Start by initializing the input image number, denoted by `i`.

Step 2: Convert `i` into a string format, resulting in `I`.

Step 3: Concatenate `I` with the image file extension (e.g., `.jpg`) to form the file name `I1`, which can now be used to access each image.

2. Read Image

Step 1: Load the image from the file `I1`. This image represents the retinal scan that will undergo classification.

3. Preprocessing

Step 1: Crop the loaded image (`I2`) to focus on the area of interest. Here, `I3` is the cropped version, defined by a four-element position vector with:

- `Xmin=50`
- `Ymin=45`
- `Width=150`
- `Height=120`

Step 2: Convert `I3` to a binary image at a threshold level of `0.45`, resulting in `I4`. This binary image highlights features like microaneurysms and haemorrhages, which will assist in identifying diabetic retinopathy stages. The image `I4` can then be plotted for verification.

4. Scanning Process

Step 1: Determine the size `[j, k]` of `I4`.

Step 2: Initialize a counter `c` to zero, which represents the count of black pixels in the binary image (the darker areas indicating abnormalities).

Step 3: Loop through each row `j` (from 1 to 120) and each column `k` (from 1 to 150):

- If the pixel at position `[j, k]` in `I4` is zero (black), increment `c` by 1.
- If the pixel is non-zero, `c` remains the same.

Step 4: Repeat this process for the entire image to obtain the final count of `c`, which will serve as a primary feature for classification.

5. Classification Based on Pixel Count

Step 1: Use the value of `c` to make a preliminary classification:

- If `c > 5000`, the detected abnormality is classified as haemorrhage, indicating a severe case.

Step 2: If `c` does not indicate haemorrhage, convert `I3` into a grayscale image (`I5`) for further analysis, and plot `I5` for reference.

6. Neural Network Initialization

Step 1: Calculate the mean(`m`) and standard deviation (`s`) of pixel intensity values in the grayscale image `I5`.

Step 2: Define `p` as a feature vector containing `[c, m, s]` to represent key features of the image.

Step 3: Initialize a feedforward neural network with activation functions `transit` (for hidden layers) and `purelin` (for the output layer).

Step 4: Set training parameters:

`Epochs = 1000`

`Goal = 0.01`

Step 5: Train the neural network with the dataset.

Step 6: Simulate the neural network with the input vector `p`.

Step 7: Multiply the simulated output by 10 and round off to approximate the classification value.

7. Final Classification

Step 1: Based on the output from the neural network:

If the output value is greater than 1, classify the type of diabetic retinopathy as Exudates.

If the output value is less than or equal to 1, classify the image as Normal.

This neural network-based classification can be refined to distinguish between Normal, Mild, Moderate, Severe, and Proliferative stages, leveraging additional statistical metrics and morphological patterns derived from the feature vector `p`.

CHAPTER 5

5. XCEPTION MODEL

5.1. Data Acquisition

The first crucial step in developing a diabetic retinopathy classification model is acquiring a large, well-annotated dataset of retinal fundus images. These images are essential as they contain detailed visual information on retinal conditions, allowing the model to learn and distinguish different stages of diabetic retinopathy (DR). Several public datasets are widely used in DR research, with APTOS (Asia Pacific Tele-Ophthalmology Society) being one of the most common. The APTOS dataset includes thousands of retinal fundus images, each labelled according to the severity of DR. Each image is classified into categories ranging from no DR (healthy retina) to the progressively severe stages of DR: mild, moderate, severe, and proliferative. Access to a well-labelled and diverse dataset is key for training the model to accurately identify varying degrees of retinal abnormalities. Such datasets often come with high-quality, expert-verified annotations, which improve the reliability and robustness of the model in clinical settings.

By acquiring a comprehensive and well-annotated dataset like APTOS, the project benefits from a structured collection of images classified by DR severity. This ensures the model has diverse examples to learn from, increasing its ability to generalize and perform accurately on new images.

5.2. Data Preprocessing

Preprocessing the retinal images is vital to prepare them for input into the Xception model or any other convolutional neural network (CNN). Preprocessing involves several key steps:

Resizing: All images are resized to a uniform dimension of 299 x 299 pixels. This standardization is important for deep learning models, as it ensures that each image has the same spatial dimensions, reducing computational load and making batch processing more efficient.

Normalization: Pixel values are scaled to a range of 0 to 1, converting the image data to a standardized format. Normalization prevents any single pixel’s intensity from dominating the model's calculations, ensuring a balanced representation of pixel values across images.

Noise Reduction: Retinal fundus images often contain noise, which can be due to lighting conditions, camera imperfections, or patient movement during image capture. Applying noise reduction techniques, such as Gaussian blur or median filtering, helps smoothen the images and removes irrelevant noise, which can otherwise interfere with feature detection.

Contrast Enhancement: Enhancing contrast improves the visibility of important details, such as blood vessels, microaneurysms, and exudates. Techniques like histogram equalization or adaptive contrast enhancement can bring out these features more prominently, making it easier for the model to recognize patterns indicative of different DR stages.

Data Augmentation: To increase the robustness of the model and prevent overfitting, data augmentation is applied to the training images. Augmentation techniques include rotation, horizontal and vertical flipping, zooming, and shearing. By artificially expanding the dataset, the model is exposed to varied perspectives and distortions of the retinal images, enhancing its ability to generalize across real-world scenarios where image orientations and qualities vary. The result of this preprocessing is a clean, consistent, and balanced dataset that is well-suited for model training. By ensuring uniform image sizes, normalized pixel values, and enhanced contrast, the pre-processed dataset allows the model to focus on relevant features without being distracted by inconsistencies or noise. Furthermore, the augmented dataset introduces variability, which increases the model's accuracy and adaptability in classifying unseen images of diabetic retinopathy, providing a reliable foundation for robust model training.

Data Annotation and Labelling

For the Xception model, which specializes in classification tasks, each retinal image is labelled according to its diabetic retinopathy severity. Categories typically include no DR, mild, moderate, severe, and proliferative stages. If applicable, additional annotations may be added to mark specific lesions, like microaneurysms or haemorrhages, to enhance training.

The result is an annotated dataset with clear labels indicating the stage of diabetic retinopathy, making it ready for supervised learning in a classification model.

5.3. Model Selection and Architecture (Xception):

The Xception model, known for its depth wise separable convolutions, is selected due to its efficiency in extracting detailed features from images. This pre-trained model serves as the backbone, providing a strong starting point. The output layer is modified to match the number of diabetic retinopathy categories (e.g., five stages), enabling the model to classify images accurately. The result is a well-defined and customized model architecture, optimized for training on the labelled retinal dataset and suited for classifying different stages of diabetic retinopathy.

5.4. Model Training Description:

To effectively train the Xception model for classifying diabetic retinopathy stages, the dataset is first split into three subsets: training (70%), validation (15%), and test (15%). This split ensures that the model has ample data to learn from, while also allowing for unbiased evaluation and fine-tuning on unseen data. The training set is used to teach the model by feeding it labelled images so it can learn to distinguish between the different stages of diabetic retinopathy. The validation set plays a key role in monitoring performance during training, allowing for adjustments and improvements. By observing how the model performs on the validation set, we can make data-driven decisions about important hyperparameters, such as the learning rate (which controls the speed of model adjustments), batch size (the number of images processed before updating model parameters), and number of epochs (complete passes over the training data).The Xception model, pre-trained on a large dataset, is fine-tuned to perform classification specific to diabetic retinopathy using categorical cross-entropy loss, a metric well-suited for multi-class classification tasks. During training, this loss function helps quantify the difference between the model's predictions and actual labels, guiding the model to improve accuracy by minimizing this difference. Hyperparameter optimization is conducted based on validation performance, ensuring that the model's settings are tailored for the best possible results on the retinal image data. Finally, after fine-tuning, the model is evaluated on the test set to measure its accuracy in classifying images into the five diabetic retinopathy stages: no DR, mild, moderate, severe, and proliferative. This process results in a highly trained model capable of distinguishing between varying stages of diabetic retinopathy, ready for practical application in medical imaging analysis with reliable accuracy.

5.5. Model Evaluation Description:

Model evaluation is a crucial step in assessing the performance of a machine learning model, especially when it comes to classifying the different stages of diabetic retinopathy (DR). Several performance metrics are used to measure how well the model is performing. Accuracy is one of the most basic metrics, representing the proportion of correct predictions (both true positives and true negatives) out of all predictions. However, accuracy alone may not be enough, especially in cases of imbalanced datasets, which is common in medical image classification. Precision and Recall are more informative in such cases. Precision measures the proportion of true positive predictions out of all positive predictions made by the model, whereas recall measures how well the model captures all actual positive cases, which is vital for medical applications where missing a positive case (e.g., undiagnosed diabetic retinopathy) can have serious consequences. The F1-score provides a balance between precision and recall, being the harmonic mean of the two. This metric is especially useful when both false positives and false negatives need to be minimized. A confusion matrix is also invaluable in visualizing the performance of the model, providing a detailed breakdown of how many instances were correctly or incorrectly classified for each class, which in the case of DR would represent the various stages of the disease.

To further ensure the robustness of the evaluation and prevent overfitting, k-fold cross-validation is often employed. This technique divides the dataset into k subsets (or "folds"), training the model on k-1 folds and testing it on the remaining fold. This process is repeated k times, with each fold being used as the test set once. The average performance across all folds provides a more reliable measure of the model's generalizability and ensures that the evaluation is not biased by a specific training/test split. Typically, values of k are set to 5 or 10, depending on the dataset size and requirements.

The outcome of this evaluation process includes detailed insights into the model's accuracy, precision, recall, F1-score, and the confusion matrix, which together highlight how well the model distinguishes between the stages of diabetic retinopathy. Additionally, cross-validation results offer further validation of the model's performance across different data subsets, ensuring it is not overfitting and can generalize well to unseen data. This comprehensive evaluation provides valuable feedback for model improvement, such as identifying areas where the model struggles with certain stages of DR, which can guide the next steps in model refinement and optimization.

5.6. XCEPTION ARCHITECTURE OVERVIEW

Input Image:

The input to the model is a retinal fundus image, which is an essential diagnostic tool for detecting diabetic retinopathy (DR) and other retinal diseases. These images provide a detailed view of the retina, allowing for the identification of various features like blood vessels, lesions, and microaneurysms. However, before feeding these images into the model, preprocessing steps are necessary to enhance the visibility of these crucial features. This preprocessing typically involves several techniques such as contrast adjustment, histogram equalization, and noise reduction. These techniques help highlight the structures in the retina, ensuring that the model can effectively learn to detect important signs of DR, such as haemorrhages, exudates, and microaneurysms.

Initial Convolutional Backbone:

The model begins with a series of convolutional layers that serve as the backbone for feature extraction from the pre-processed retinal fundus images. The convolutional layers apply filters to the input image to detect patterns, edges, and textures.

First Convolutional Layer: This layer applies 32 filters of size 3×3 (kernel size), with a stride of 2. The stride of 2 reduces the spatial dimensions of the image, effectively down sampling it. This layer uses ReLU activation, which introduces non-linearity to the model and helps it learn more complex patterns in the data.

Second Convolutional Layer: This layer uses 64 filters with the same 3×3 kernel but a stride of 1. The stride of 1 ensures that the output retains more of the spatial resolution, allowing the model to learn finer details in the image. As with the first layer, ReLU activation is applied to introduce non-linearity and enable the model to learn more complex representations. These initial convolutional layers are designed to capture low-level features, such as edges, textures, and simple shapes. These features are the building blocks for more complex representations that are learned in later layers.

Depth wise Separable Convolutions (Xception Backbone):

The heart of the model uses the Xception architecture, which is an extension of the traditional Inception architecture that focuses on depth wise separable convolutions.

This approach is computationally more efficient and helps in learning more effective feature representations, especially in the context of image classification tasks.

Depth wise Convolution: In this type of convolution, a single convolutional filter is applied to each input channel separately. For example, in an RGB image, there are three channels, and each channel is processed independently. This allows for the model to capture spatial features in each channel without mixing them prematurely, which reduces computational complexity.

Pointwise Convolution: Following the depth wise convolution, a 1×1 convolution is applied across all channels to mix the features extracted by the depth wise convolution. This allows the model to combine information across different channels, enhancing the feature representation without requiring as many parameters as standard convolutions.

Each block in the Xception architecture contains the following components:

Batch Normalization: This helps in stabilizing the learning process by normalizing the activations of the previous layer, ensuring they don't get too large or small, which could hinder training.

ReLU Activation: As with earlier layers, ReLU activation is used to introduce non-linearity and enable the model to learn complex patterns.

Stack of Depth wise Separable Convolution Blocks:

The Xception architecture consists of a series of depth wise separable convolution blocks, organized into three main flows:

Entry Flow: The initial blocks of the architecture capture high-level features in the input image. These blocks begin to extract more abstract patterns from the retinal fundus image, such as the shape and structure of blood vessels and lesions.

Middle Flow: This part consists of several depth wise separable convolution blocks (typically 8). The middle flow deepens the feature extraction process, learning more intricate patterns and representations of the retina, such as specific characteristics of diabetic retinopathy.

Exit Flow: The final blocks in the Xception model reduce the spatial dimensions of the learned features. This helps in preparing the features for final classification by compressing the learned spatial information into a lower-dimensional representation, making the model more efficient.

Global Average Pooling Layer:

After the final convolution block, a global average pooling layer is used. This layer reduces the spatial dimensions of the feature maps from the convolution blocks into a single vector per channel, effectively summarizing the information across the entire image. The pooling layer computes the average of each channel, ensuring that the model retains global information while reducing the complexity of the feature maps.

Fully Connected Layer:

The output from the global average pooling layer is passed through one or more fully connected layers (also known as dense layers). These layers are responsible for combining the high-level features extracted by the convolutional layers and making the final classification decision. The fully connected layers allow the model to make predictions based on the learned features.

CHAPTER 6

SAMPLE CODE

```
import os

import numpy as np

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import Xception

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

train_dir = 'data'  # Your training data directory

class_names = sorted(os.listdir(train_dir))  # Get class names from folder names

label_encoder = LabelEncoder()

label_encoder.fit(class_names)

num_classes = len(class_names)

train_datagen = ImageDataGenerator(

    rescale=1.0/255,

    rotation_range=20,

    width_shift_range=0.2,

    height_shift_range=0.2,

    shear_range=0.2,

    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(299, 299),  # Resize for Xception input
    batch_size=32,
    class_mode='categorical'
)
def build_xception_model():
    base_model = Xception(weights='imagenet', include_top=False, 
input_shape=(299, 299, 3))
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    return model
def build_xception_model():
    base_model = Xception(weights='imagenet', include_top=False, 
input_shape=(299, 299, 3))
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    return model
def compile_and_train(model, model_name):
    model.compile(optimizer=Adam(learning_rate=1e-4),
loss='categorical_crossentropy', metrics=['accuracy'])
     # Define Callbacks
    checkpoint          =          ModelCheckpoint(f'{model_name}_best_model.keras',
monitor='val_accuracy', save_best_only=True, mode='max')
    early_stopping        =        EarlyStopping(monitor='val_loss',        patience=5,
restore_best_weights=True)
     # Train the model
    model.fit(
        train_generator,
        epochs=20,
        callbacks=[checkpoint, early_stopping]
    )
```

CHAPTER 7

SYSTEM TESTING

7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS

7.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2 Integration testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is cantered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot <see= into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.5 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

▪ All field entries must work properly.

▪ Pages must be activated from the identified link.

▪ The entry screen, messages and responses must not be delayed.

Features to be tested

▪ Verify that the entries are of the correct format

▪ No duplicate entries should be allowed

▪ All links should take the user to the correct page.

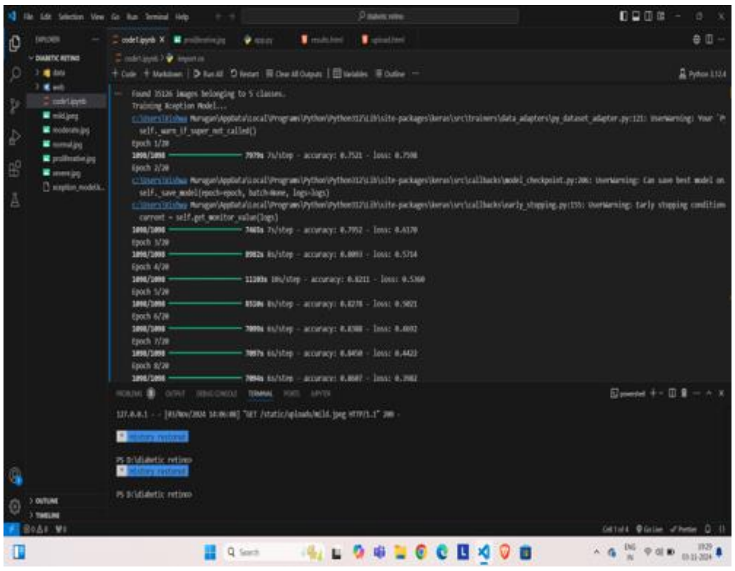
Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level –interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered. Test Environment Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested. In our project we aim to predict the plant diseases using their leaves. And we will classify the leaf with their diseases, details of that disease and solution.

Unit Testing Of Modules Unit testing involves the design of test cases that validate that the Internal program logic is functioning properly and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software unit of the application; it is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and its invasive. Unit tests perform basic test at component level and test the specific business process application and/or system configuration. Unit tests ensured that each unit path of a business process performs accurately to the documented specification and contains clearly defined input and expected results.

CHAPTER 8

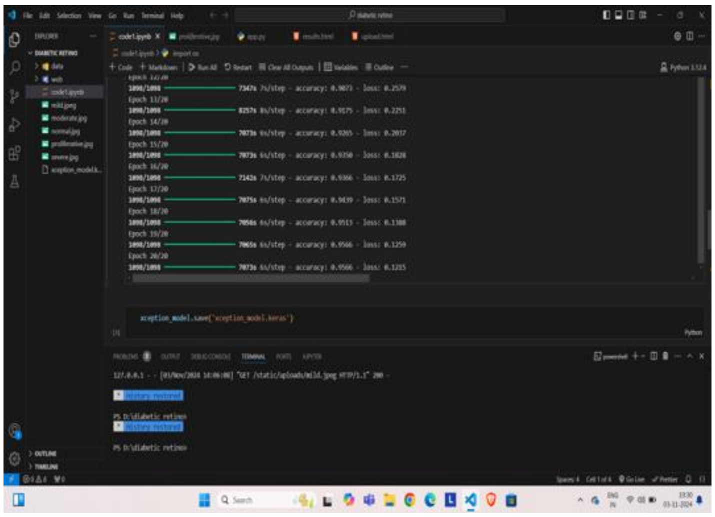
8.RESULT AND CONCLUSION



```
Found 35126 images belonging to 5 classes.
Training Xception Model...
c:\Users\Vishwa Murugan\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `P
  self._warn_if_super_not_called()
Epoch 1/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7979s 7s/step - accuracy: 0.7521 - loss: 0.7598
Epoch 2/20
c:\Users\Vishwa Murugan\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\callbacks\model_checkpoint.py:206: UserWarning: Can save best model on
  self._save_model(epoch=epoch, batch=None, logs=logs)
c:\Users\Vishwa Murugan\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\callbacks\early_stopping.py:155: UserWarning: Early stopping condition
  current = self.get_monitor_value(logs)
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7461s 7s/step - accuracy: 0.7952 - loss: 0.6170
Epoch 3/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 8982s 8s/step - accuracy: 0.8093 - loss: 0.5714
Epoch 4/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 11103s 10s/step - accuracy: 0.8211 - loss: 0.5360
Epoch 5/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 8510s 8s/step - accuracy: 0.8278 - loss: 0.5021
Epoch 6/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7099s 6s/step - accuracy: 0.8380 - loss: 0.4892
Epoch 7/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7097s 6s/step - accuracy: 0.8450 - loss: 0.4422
Epoch 8/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7094s 6s/step - accuracy: 0.8587 - loss: 0.3982
```

```
127.0.0.1 - - [01/Nov/2024 14:06:08] "GET /static/uploads/mild.jpeg HTTP/1.1" 200 -

 *  History restored

PS D:\diabetic retino>
 *  History restored

PS D:\diabetic retino>
```

9.Training Model of 20 epoch

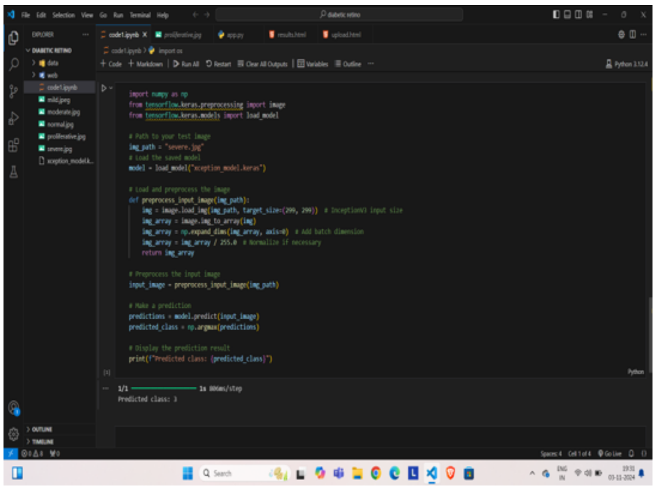


```
Epoch 12/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7347s 7s/step - accuracy: 0.9073 - loss: 0.2579
Epoch 13/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 8257s 8s/step - accuracy: 0.9175 - loss: 0.2251
Epoch 14/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7073s 6s/step - accuracy: 0.9265 - loss: 0.2017
Epoch 15/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7073s 6s/step - accuracy: 0.9350 - loss: 0.1828
Epoch 16/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7142s 7s/step - accuracy: 0.9366 - loss: 0.1725
Epoch 17/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7075s 6s/step - accuracy: 0.9439 - loss: 0.1571
Epoch 18/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7058s 6s/step - accuracy: 0.9513 - loss: 0.1388
Epoch 19/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7065s 6s/step - accuracy: 0.9566 - loss: 0.1259
Epoch 20/20
1098/1098 ━━━━━━━━━━━━━━━━━━━━ 7073s 6s/step - accuracy: 0.9566 - loss: 0.1215
```

```python
xception_model.save('xception_model.keras')
```

```
127.0.0.1 - - [01/Nov/2024 14:06:08] "GET /static/uploads/mild.jpeg HTTP/1.1" 200 -

 *  History restored

PS D:\diabetic retino>
 *  History restored

PS D:\diabetic retino>
```

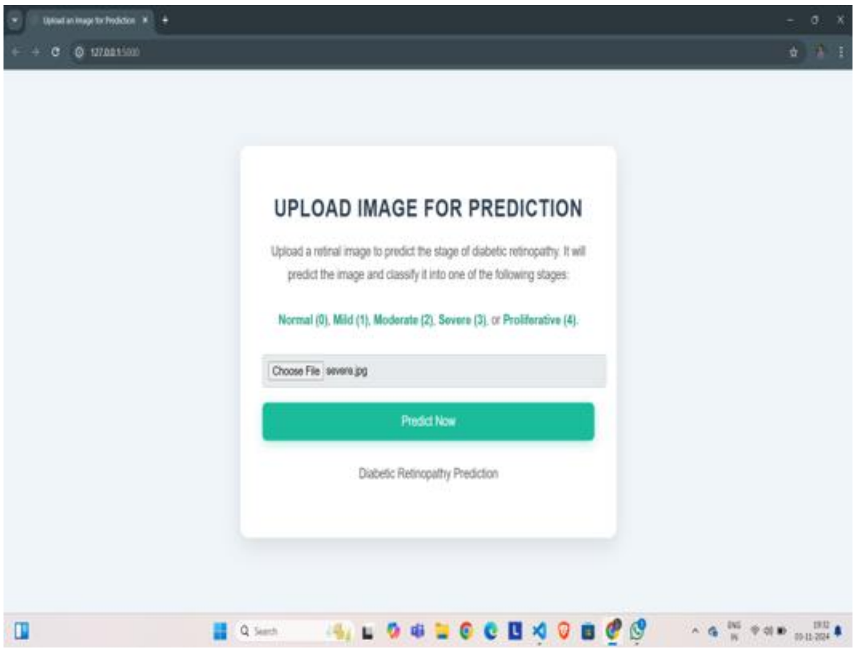
10. 95.66% accuracy of final epoch



```python
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model

# Path to your test image
img_path = "severe.jpg"
# Load the saved model
model = load_model("xception_model.keras")

# Load and preprocess the image
def preprocess_input_image(img_path):
    img = image.load_img(img_path, target_size=(299, 299))  # InceptionV3 input size
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    img_array = img_array / 255.0  # Normalize if necessary
    return img_array

# Preprocess the input image
input_image = preprocess_input_image(img_path)

# Make a prediction
predictions = model.predict(input_image)
predicted_class = np.argmax(predictions)

# Display the prediction result
print(f"Predicted class: {predicted_class}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 886ms/step
Predicted class: 3
```

11. Direct Output



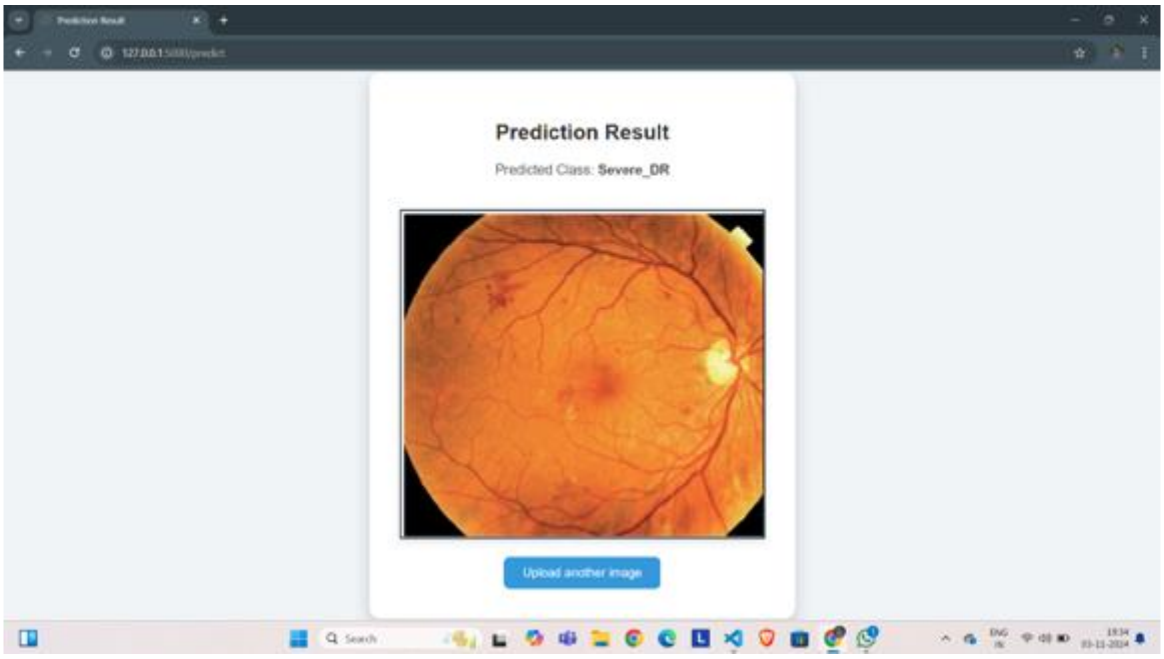
UPLOAD IMAGE FOR PREDICTION

Upload a retinal image to predict the stage of diabetic retinopathy. It will predict the image and classify it into one of the following stages:

Normal (0), **Mild (1)**, **Moderate (2)**, **Severe (3)**, or **Proliferative (4)**.

Diabetic Retinopathy Prediction

12. Web Output



Prediction Result

Predicted Class: **Severe_DR**

13. Web Output

CONCLUSION

In this project, we developed a highly effective diabetic retinopathy (DR) prediction model using the Xception deep learning architecture. The model achieved an impressive accuracy of 95.66%, demonstrating its ability to accurately identify and classify various stages of diabetic retinopathy, ranging from no DR to proliferative DR. Xception's architecture, known for its depth wise separable convolutions, allowed the model to extract high-quality features from retinal fundus images with reduced computational costs, making it both efficient and effective. This makes it a promising tool for early detection and diagnosis of diabetic retinopathy, which is crucial for preventing vision loss in diabetic patients.

To further enhance the model's performance, we leveraged transfer learning by using a pre-trained Xception model. This approach significantly accelerated the training process and improved the model's accuracy, even when the available labelled data was limited. Transfer learning allowed the model to benefit from features learned on large-scale datasets, making it more robust and capable of generalizing well to the specific task of diabetic retinopathy classification. Overall, the success of this project highlights the potential of deep learning models like Xception for medical image analysis and the early detection of eye-related diseases, offering a valuable tool for healthcare professionals.

CHAPTER 9

REFERENCES

1.Prediction of Diabetic Retinopathy Based on Risk Factors using Machine Learning Algorithms/2023 4th International Conference on Artificial Intelligence and Data Sciences (AIDAS)/DOI: 10.1109/AIDAS60501.2023.10284646

2. Diabetic Retinopathy Prediction based on CNN and Alex Net model/2024 14th InternationalConferenceonCloudComputing,DataScience&amp;Engineering(confluence)/DOI:10.1109/CONFLUENCE60223.2024.10463351

3. Detection of Diabetic Retinopathy with Retinal Images using CNN/Sixth International Conference on Intelligent Computing and Control Systems (ICICCS 2022)/DOI:10.1109/ICICCS53718.2022.9788368

4.Detection and Classification of Diabetic Retinopathy using Pretrained Deep Neural Networks/2023 International Conference on Innovation in EngineeringandTechnology(ICIET)/DOI:10.1109/ICIET57285.2023.10220715

5. Classification of Diabetic Retinopathy Using Image Pre-processing Techniques/2023 3rd International Conference on Intelligent Technologies (CONIT) /DOI:10.1 109/CONIT59222.2023.10205586