Enhanced Optimized CNN Based Automated Diabetic Retinopathy Detection

A Project Report submitted in the partial fulfillment of the

Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Vankayala Teja Sri (21471A05D3)

Akkapalli Venkayamma (21471A05D8)

Karna Eswar Kalyani (21471A0594)

Under the esteemed guidance of

Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.,

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Tire -1 NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name **"Enhanced Optimized CNN Based Automated Diabetic Retinopathy Detection"** is a Bonafide work done by the team **Vankayala Teja Sri** (21471A05D3), **Akkapalli Venkayamma** (21471A05D8), **Karna Eswar Kalyani** (21471A0594) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE
Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.,
Associate Professor

PROJECT CO-ORDINATOR
Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.,
Associate Professor

HEAD OF THE DEPARTMENT
Dr. S. N. Tirumala Rao, B.Tech., M.Tech., Ph.D.,
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "**Enhanced Optimized CNN Based Automated Diabetic Retinopathy Detection**" is composed by ourselves that the work contained here is our own except where explicitly stated otherwise in the text and that this work has not been submitted for anyother degree or professional qualification except as specified.

By

Vankayala Teja Sri (21471A05D3)
Akkapalli Venkayamma (21471A05D8)
Karna Eswar Kalyani (21471A0594)

ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V.Koteswara Rao**, **B.Sc.**, who took a keen interest in us in every effort throughout this course.We owe our sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao, B.Tech., M.Tech., Ph.D.,** HOD of CSE department and also to our guide **Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.,** of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks to **Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.,** Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant sourceof inspiration for us throughout this project work.

We extend our sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration, and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends andthose who were involved in giving valuable suggestions clarified our doubts, which really helped us in successfully completing our project.

By

Vankayala Teja Sri (21471A05D3)
Akkapalli Venkayamma (21471A05D8)
Karna Eswar Kalyani (21471A0594)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for researchand ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern softwaretools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and

skills in numerous areas of Computer Science and Engineering to design and develop software-basedsystems.

PSO2: Acquaint module knowledge on emerging trends of the modern era inComputer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems relatedto academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and cancommunicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineeringfundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, research literature, and analyse complexengineering problems reaching substantiated conclusions using first principles ofmathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge toassess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional solutions in societal and environmental contexts, and demonstrate the knowledge of, andneed for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilitiesand norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member orleader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and giveand receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of theengineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.**CO421.3:** Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		√											√		
C421.2	√		√		√								√		
C421.3				√		√	√	√					√		
C421.4			√			√	√	√					√	√	
C421.5					√	√	√	√	√	√	√	√	√	√	√
C421.6									√	√	√		√	√	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO 2	PSO 3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlationbetween CO's and PO's:

1. Low level

2. Medium level

3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are appliedin this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, planning to develop a Deep Learning model for Diabetic Retinopathy detection using CNN and retinal image analysis.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process modelis identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3,C3204.3, C4110.2	Implementation is done, and healthcare professionals and patients will use the project. Future updates can include early-stage DR and additional eye disease screening.	PO4, PO7
C32SC4.3	The physical design includes a web application to analyze retinal images and predict DR stages using Deep Learning	PO5, PO6

ABSTRACT

Early identification is essential to prevent serious visual impairment in diabetic patients as Diabetic Retinopathy (DR) is the main reason for blindness. In this project, an optimal Convolutional Neural Network (CNN) model is used to propose an automated approach for categorizing the stages of DR. The pre-trained VGG16 model uses deep feature extraction for the given retinal pictures, which uses scaling and feature selection heuristics by the Gray Wolf Optimizer. Those selected features from GWO belonging to the most relevant features would also give a better boost to the classification performance along with the optimization of both hyperparameters. The proposed model will perform better than the traditional techniques based on the Precision, Recall, and F1 scores' experimental results. It has an accuracy of 99.31% on the DR dataset. The inclusion of GWO in CNN, models holds tremendous potential for use in the analysis of medical images and yields Optimized CNN, an efficient technique that is effective in improving healthcare diagnosis.

INDEX

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Diabetic Retinopathy (DR) is one of the most severe complications of diabetes, often leading to vision impairment and blindness if not diagnosed early. It occurs due to prolonged high blood sugar levels, which damage the blood vessels in the retina, leading to leakage, swelling, or abnormal blood vessel growth. DR progresses through different stages, from mild non-proliferative retinopathy to severe proliferative retinopathy, where abnormal vessel growth can cause retinal detachment and permanent vision loss. Early detection and timely intervention are crucial in preventing irreversible blindness.

DR prevalence varies based on age, gender, and duration of diabetes as shown in Fig .1.1. Studies indicate that older adults (60+ years) are at a higher risk, with DR affecting nearly 18-20\% of diabetic individuals in this age group. However, working-age adults (40-59 years) also experience significant DR incidence, making it a major cause of vision impairment in economically active populations. Gender-based studies suggest that while men are more likely to develop severe DR, women have a higher overall prevalence, possibly due to hormonal changes and gestational diabetes [1]. Moreover, individuals who have had diabetes for more than 10-15 years are significantly more prone to DR. These demographic insights emphasize the importance of targeted screening and early intervention strategies.

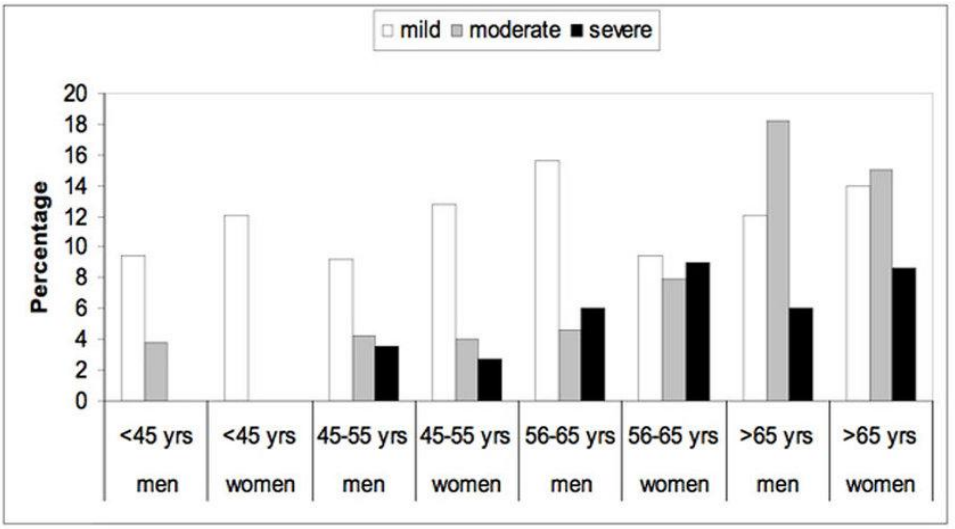

Fig 1.1 Prevalence of Diabetic Retinopathy by Age, Gender, and Severity

Traditional methods for DR detection rely on manual assessment by ophthalmologists, which can be time-consuming, prone to human error, and inaccessible in remote areas. To address these challenges, Deep Learning and artificial intelligence (AI) have emerged as powerful tools in automating DR detection [2], improving diagnostic efficiency, and enhancing patient outcomes.

Deep Learning-based methods, particularly Convolutional neural networks (CNNs), have revolutionized medical image analysis by enabling automated feature extraction and classification. Several Deep Learning architectures, such as VGG16, ResNet, and DenseNet [3], have demonstrated significant success in medical imaging tasks, including DR detection. However, challenges such as imbalanced datasets, feature redundancy, and hyperparameter optimization still hinder their full potential. Addressing these limitations requires an optimized approach that enhances both feature selection and model accuracy.

To improve upon existing methods, this study explores a hybrid Deep Learning approach that integrates CNNs with optimization techniques [4]. Specifically, it employs the Gray Wolf Optimizer (GWO) for feature selection and hyperparameter tuning, ensuring that only the most relevant features contribute to classification performance. By refining feature selection and reducing computational complexity, this approach aims to enhance DR detection accuracy while maintaining efficiency.

Furthermore, recent advancements in AI-based DR detection have led to the development of hybrid models that combine multiple architectures for improved performance. Techniques such as transfer learning [5], attention mechanisms, and adaptive optimization algorithms have shown promise in increasing sensitivity and specificity. This research investigates how integrating such methodologies with CNNs can lead to a more robust and scalable solution for DR diagnosis.

The significance of this study lies in its potential to bridge the gap between research advancements and real-world clinical applications. By leveraging AI-driven methods, healthcare professionals can access reliable, automated tools for early DR diagnosis, reducing the risk of vision loss in diabetic patients.

Additionally [12], the use of optimization techniques like GWO ensures that the model achieves high accuracy without excessive computational demands. The remainder of this paper is structured as follows: Section II reviews related work on DR detection using Deep Learning models and optimization techniques. Section III describes the proposed methodology, including dataset preprocessing, feature extraction, and model optimization. Section IV presents experimental results and performance evaluation. Finally, Section V discusses the study's conclusions and future research directions.

2. LITERATURE SURVEY

2.1 RELATED WORK

Previous research on the identification of Diabetic Retinopathy is compiled in the related work section, with an emphasis on several Deep Learning models, including VGG, ResNet, and DenseNet. It highlights the inadequacies of these current methods in obtaining high sensitivity and accuracy, which justifies the need for the optimized Convolutional Neural Network (OpCoNet) in conjunction with Gray Wolf Optimization (GWO) to improve detection efficacy.

Using Deep Learning algorithms,[1] Shankar et al. were able to diagnose Diabetic Retinopathy with 99.28\% accuracy. Other research found that ResNet34 produced 86\% sensitivity and 85% accuracy, decision tree-based systems produced 91\% accuracy, and VGGNet produced 92% accuracy. These results demonstrate how Deep Learning models can improve ophthalmic diagnostic accuracy.

Gadekallu et al. used a Deep Learning model with efficient feature extraction to predict Diabetic Retinopathy (DR) with 96% accuracy. Concerning automated DR detection,[2] their study demonstrated the effectiveness of Convolutional neural networks (CNNs) in image processing.

Gaurav Saxena optimized hyperparameters with a batch size of 16 and a learning rate of 0.001 to create a deep-learning model for Diabetic Retinopathy diagnosis using InceptionResNetV2. CNNs have potential in clinical contexts and are successful for [3] automated DR screening, as demonstrated by the model's Accuracy of 92% on the Messidor-2 dataset.

Ayesha Jabbar and associates achieved 94% accuracy with a hybrid Deep Learning model that combined ResNet and GoogleNet with an adaptive particle swarm [4] optimizer. To improve performance, they employed preprocessing methods, transfer learning, and several classifiers. They measured the results using measures including F1 score, accuracy, precision, and recall. Demographic biases in the datasets were also examined in the study.

Using CNNs, Muhammad Shoaib Farooq diagnosed Diabetic Retinopathy (DR) with 73% accuracy. Raza Shah et al.'s hybrid CNN-RNN [5] model achieved 90% accuracy, whereas Aamir Ali et al. used hyperparameter optimization and transfer learning to increase accuracy to 85%. By using cross-validation and parameter optimization, Muhammad Younas Javed et al. were able to obtain 88% accuracy, highlighting the potential of Deep Learning to automate DR diagnosis and enhance results.

Using the EyePACS dataset, Muhammad Mohsin Butt's modified VGGNet CNN was able to diagnose Diabetic Retinopathy with 83.1% accuracy. D. N. F. Awang Iskandar et al. used a Convolutional attention module with a DenseNet encoder to achieve 82% multiclass and 97% binary classification accuracy [6]. Deep Learning is effective in DR identification, as demonstrated by Ghazanfar Latif et al. who achieved 94% accuracy in binary classification by augmenting feature extraction with artificial synaptic meta-plasticity.

With an Integrated Brier Score (IBS) ranging from 0.049 to 0.161, Bin Sheng's DeepDR Plus system was able to predict Diabetic Retinopathy with C-index values of 0.823 to 0.862. With C-index values of 0.794 to 0.842 in external validation, the study, which involved Huating Li et al.,[7] highlighted customized screening intervals and showed the model's efficacy in early DR detection and management.

The DR-ResNet+ model developed by Gaurav Dhiman et al. achieved 98.98% accuracy, 98.29% sensitivity, and 99.16% specificity after being optimized with grid and random search for hyperparameters. Verified using VGG16, AlexNet, and GoogleNet,[8] it shows great promise for early detection and treatment of Diabetic Retinopathy.

Using a two-pronged strategy for Diabetic Retinopathy classification, B. N. Jagadesh et al.'s IC2T model [9] obtained 96%, 97%, and 98% accuracy rates on the EyePACS-1, Messidor-2, and DIARETDB0 datasets. By emphasizing blood vessels, optic discs, and retinal biomarkers, their approach highlights the significance of model architecture and parameter optimization in improving DR detection.

Using transfer learning with InceptionV3 and an improved custom classification layer,[10] Fathi Kallel et al. achieved 96.88% accuracy in the categorization of Diabetic Retinopathy. Wan et al. used transfer learning with multiple architectures to reach 95.68% accuracy, whereas Hemanth et al. used a CNN model and enhanced visual contrast to get 97% accuracy.

2.2 APPLICATIONS OF DEEP LEARNING

1. Image Recognition and Classification
2. Natural Language Processing (NLP)
3. Speech Recognition and Synthesis
4. HealthCare
5. Finance
6. Autonomous Systems
7. Recommendation Systems
8. Robotics
9. Image Captioning
10. Education and Research
11. Self-Driving Cars
12. Natural Language Processing
13. Climate and Environment Science
14. Fraud Detection
15. Detecting Developmental Delay in Children
16. Colourisation of Black and White images
17. Manufacture and Industry
18. Automatic Machine Translation
19. Generative Modeling

3. EXISTING SYSTEM

Manual Feature Extraction and Selection:

Traditional methods for feature extraction in image classification rely on manual selection techniques such as Principal Component Analysis (PCA) and Local Binary Patterns (LBP). These approaches are often suboptimal as they require domain expertise and may not capture the most informative features, limiting classification performance.

Conventional CNN Models Without Optimization:

Standard Convolutional Neural Networks (CNNs) process all extracted features from images without any feature selection, leading to high computational costs and overfitting risks. These models demand extensive training time and large datasets to generalize effectively, making them resource-intensive.

Feature Selection Using Traditional Methods:

Some prior systems use basic feature selection techniques such as Recursive Feature Elimination (RFE) or Mutual Information-based selection. However, these methods do not dynamically adapt to different datasets and may discard important features, affecting model accuracy.

Lack of Optimization in Deep Learning Models:

Many Deep Learning approaches do not incorporate metaheuristic optimization techniques like Gray Wolf Optimization (GWO) for feature selection. Without optimization, models process redundant or irrelevant features, increasing computational complexity without significant accuracy gains.

Computational Inefficiencies in Large-Scale Datasets:

Traditional Deep Learning models require significant computational power to process high-dimensional image features, making them impractical for real-time applications. The absence of feature reduction mechanisms results in longer training times and inefficient memory usage.

4. PROPOSED METHODOLOGY

To enhance the efficiency and accuracy of Deep Learning based classification, this study proposes an Optimized Convolutional Neural Network (OpCoNet) that integrates Gray Wolf Optimization (GWO) for effective feature selection. Traditional CNN models process large sets of extracted features, increasing computational costs and overfitting risks. We aim to reduce feature redundancy and improve model performance by applying metaheuristic optimization techniques. The Fig. 4.1, represents the series of steps for implementing the CNN with the optimization algorithm.

- **Dataset Acquisition:** The dataset consists of retinal images categorized into different severity levels of Diabetic Retinopathy (DR). 3,662 images are used for training, ensuring a diverse representation of different DR stages. These images are pre-processed and stored for further feature extraction and model training.

- **Data preprocessing:** The data preprocessing phase involves resizing all images to 224 × 224 pixels for uniform input and normalizing pixel values between 0 and 1 to enhance convergence. Class weights address class imbalance to prevent model bias.

- **Feature Extraction**: Deep feature extraction is performed using a pre-trained VGG16 model, which captures high-level patterns from retinal images. Each image initially contains 25,088 extracted features, representing various structural and textural details crucial for DR classification. These features serve as input for the optimization process.

- **Feature Selection and Hyperparameter Tuning Using Optimization:** To reduce dimensionality and computational complexity, GWO is applied for feature selection. The GWO-based selection reduces the feature count to 10,316. This step ensures that only the most relevant features are retained, improving model efficiency and accuracy while reducing overfitting.

- **Model Construction:** The Optimized CNN (OpCoNet) is designed with multiple layers, including dense layers, dropout layers, and batch normalization to stabilize learning. The selected features serve as input to a fully connected neural network,

optimized using hyperparameter tuning techniques derived from GWO to enhance classification performance.

- **Training Process:** The CNN model is trained using the Adam optimizer with a learning rate optimized by GWO. The categorical cross-entropy loss function is used for multi-class classification. The model is trained over 25 epochs, ensuring convergence while preventing overfitting. During this phase, the selected features contribute to faster and more efficient learning.

- **Model Evaluation:** The trained model is tested on a validation dataset, evaluating key metrics such as accuracy, precision, recall, and F1-score. The GWO-selected features model achieves 99.31% accuracy, Additionally, training time is analyzed, where GWO outperforms ACO in computational efficiency.

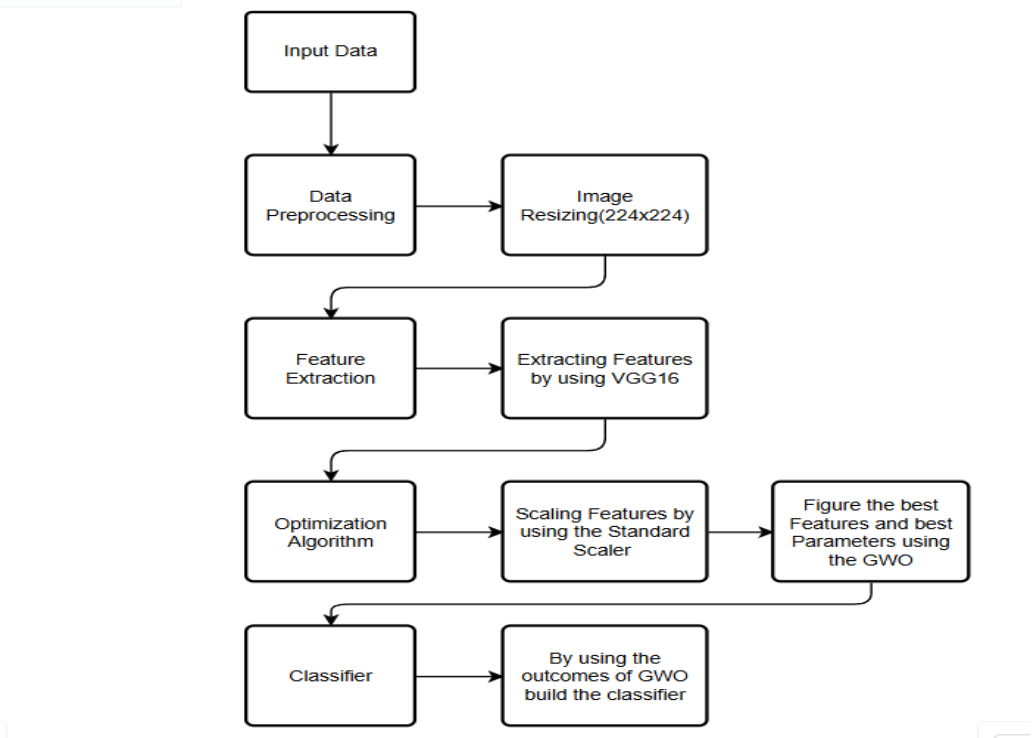

Fig 4.1 Proposed Methodology

5. SYSTEM REQUIREMENTS

5.1 Hardware Requirements:

- System Type : AMD Ryzen 5 5000 Series Processor
- Cache memory : 8MB(Megabyte)
- Graphics : NVIDIA Panel (dedicated GPU support)
- RAM : 16GB (gigabyte)
- Hard Disk : 512GB SSD

5.2 Software Requirements:

- Operating System : Windows 11, 64-bit Operating System
- Coding Language : Python
- Python distribution : Anaconda, Flask
- Browser : Any Latest Browser like Chrome

6. SYSTEM ANALYSIS

6.1 Scope of the project

Optimized Feature Selection: This project focuses on developing and applying advanced optimization techniques, such as Gray Wolf Optimization (GWO) and Ant Colony Optimization (ACO), to select the most relevant features from large datasets. By reducing the feature set while maintaining or improving model accuracy, the project aims to enhance computational efficiency and model performance.

Model Performance Enhancement: Central to this project is constructing optimized Convolutional Neural Networks (OpCoNet) that leverage the selected features to achieve high accuracy in image classification tasks. The project will explore the impact of different optimization strategies on model accuracy, training time, and overall computational efficiency.

Comparative Analysis: A key part of the project involves comparing the effectiveness of various feature selection methods, including using all extracted features versus optimized subsets. This comparison will highlight the trade-offs between accuracy and computational cost, providing insights into the best practices for feature selection in Deep Learning models.

Application to Image-Based Datasets: The project is specifically tailored to image-based datasets, analyzing how optimized CNNs perform in recognizing patterns and classifying images with high precision. While the primary focus is on image data, the findings could be adapted for other data types in future research.

Mitigating Computational Challenges: With an emphasis on reducing the computational burden, the project seeks to address challenges related to model training times and resource consumption. By optimizing feature selection, the goal is to make Deep Learning models more accessible and efficient for broader applications.

6.2 Dataset Analysis

The "Diabetic Retinopathy Level Detection" dataset on Kaggle includes retinal images categorized into different levels of severity of Diabetic Retinopathy. The present research used 4396 images to classify the level of DR. The Complete process divides the photographs into the Training and Validation sets. The Fig. 6.1, explains that the DR Photographs are categorized into 5 Levels. Diabetic Retinopathy (DR) is absent No symptoms of Diabetic Retinopathy are present at Level-0 The retina seems to be in normal condition.

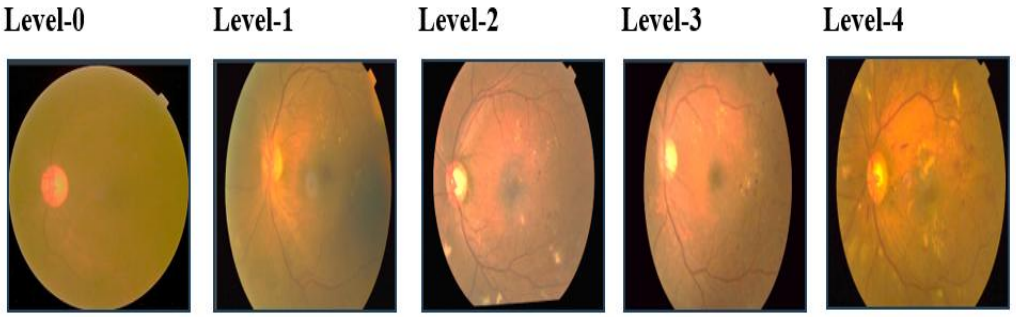

Fig 6.1 Severity Of DR

Microaneurysms and tiny blood vessel bulges in the retina characterize level 1 Mild Non-Proliferative Diabetic Retinopathy. There are no further retinopathy symptoms. Diabetic Retinopathy with Moderate Non-Proliferative Effects Level 2 There are more microaneurysms along with other abnormalities such as retinal hemorrhages and exudates. There can be some anomalies in the blood vessels as well. Severe Diabetic Retinopathy with No Proliferation Level 3 More substantial alterations are present at this level, including several hemorrhages in the retina Venous beading (veins twisted and dilated) anomalies of the intraretinal microvascular system (IRMA) The likelihood of developing proliferative Diabetic Retinopathy rises at this point. Diabetic Proliferative Retinopathy Level 4 is characterized by creating new blood vessels on the optic disc or retina (neovascularization). This stage can lead to serious complications, including vitreous hemorrhages and retinal detachment, which can cause permanent vision loss if not treated

Fig 6.2 Class imbalance of images in the training dataset.

A total of 3662 images are used for training the optimized Convolutional neural network [1]. Most images belong to no DR Stage representing the Class Imbalance. Fig. 6.2, represents the Class Imbalance of the photographs.

6.3 Data Preprocessing

The Diabetic Retinopathy Level Detection dataset is made up of high-quality images. To speed up the training process, the images were resized to 224x224 and processed in batches of 32 to ensure consistency and conformity with the VGG16 model used for feature extraction. Fig. 6.3, outlines the differences before and after preprocessing. These methods improve computational efficiency and the model's ability to identify features when classifying Diabetic Retinopathy. Since the dataset was already divided to accommodate both the training and testing sets, data splitting was not required. Class weights are computed to address an issue referred to as class imbalance [8] this prevents the model from being biased toward the majority class and thus removes the overuse of one's respective sampling of data to overcome this phenomenon. No data augmentation techniques, such as rotations and flips, were performed in the process, and instead solely relied upon the given preprocessing procedures.

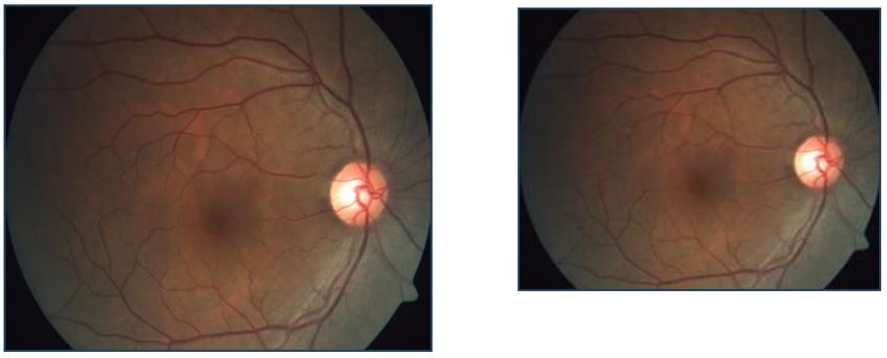

Image Before Preprocessing Image After Preprocessing

Fig 6.3 The Comparison of Images Before and After Preprocessing

6.4 Feature Extraction:

The procedure extracts deep features from a collection of photos using a pre-trained Convolutional neural network (CNN), more precisely the VGG16 model [10]. The Oxford Visual Graphics Group created VGG16, which consists of 16 layers where weights can be learned. It is well known for its simplicity and depth. Because of its capacity to extract features from images and learn high-level abstract features from them, it is commonly used for image categorization [14]. Here, we use our unique dataset to do feature extraction using the VGG16 model without completely linked layers. We have two directories containing images: one for training and another for validation. To prepare the images for input into the VGG16 model, we first normalize the pixel values to fall between 0 and 1, divided by 255.0. This normalization step helps in faster convergence during training by scaling down the pixel intensity values. We use Image Data Generator with a preprocessing function specific to VGG16, called preprocess input. This function performs the scaling and normalization required by the pre-trained VGG16 model. The data generator also provides a method to generate batches of images with their respective labels for both training and validation datasets. We then initialize two data generators. Both generators are configured to resize images to 224x224 pixels (the input size required by VGG16) and to output batches of 32 images.

The VGG16 model is loaded with pre-trained weights from the ImageNet dataset. The feature extraction process involves passing batches of images through VGG16's Convolutional base. The output is a set of deep features for each image that capture high-level patterns useful for image classification.

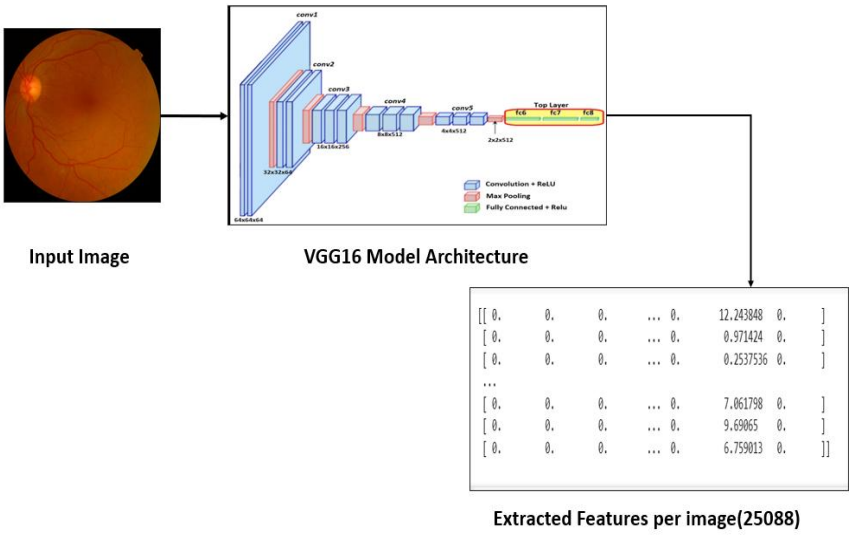

Fig 6.4 VGG16 Feature Extraction

After they have been retrieved, the features are combined into a single array as shown in Fig. 6.4. Similar processing and concatenation are applied to the labels as well. The extracted features and the labels that go with them are then saved to disk. The NumPy.save() function, which effectively writes the features and labels to .npy files, can be used for this. The computational time required for feature extraction can be greatly decreased by loading these data later on to train a classifier. We ensure that the expensive process of running images through the VGG16 model only needs to be done once by storing the extracted features, which makes it possible to build the classification model more quickly. The features that have been preserved can be utilized for many purposes, such as further refining the model or training machine learning classifiers

6.5 Gray Wolf Optimization:

The Gray Wolf Optimizer (GWO) is a nature-inspired optimization algorithm that mimics Gray wolves' social order and hunting behaviour in the wild. Mirjalili et al. introduced it in 2014. The primary purpose of GWO is to expedite the search space and compare the Gray wolf hunting process to determine the best solution to a problem. GWO does an excellent job of optimizing parameters [11-12].

Feature selection extracts the most significant features from high-dimensional datasets, simplifying and improving model performance. Global Optimization Finding the global optimum for a particular problem while avoiding local optimal traps is critical in complicated or nonlinear systems [13]. We extracted features from our dataset, resulting in a high dimensional feature space with 25,088 features per Image. Feature Scale We applied a standard scaling method to the extracted data from our training and validation sets to ensure a consistent comparison of features. To narrow down the criteria and select only the most important features, we used GWO. A feature selection mask was then applied to the scaled input data.

This mask acts as a binary filter: a value of '1' indicates that the feature is selected, while '0' means it is excluded. The decision to include or exclude each feature is guided by the GWO algorithm, which evaluates the contribution of each feature to the model's performance. Class homogeneity We calculated the class weights to check for the homogeneity of the data set so that the CNN model is not biased toward one class. The CNN model is designed with two connected layers (thick). In addition, a feature selection mask was used to select the most relevant features from the input data. The objective function assessed the model's performance in the validation set by calculating the validation accuracy. The most important optimal parameters are:

- Number of neurons in the first dense layer.
- Number of neurons in the second layer.
- Learning speed for Glasses (Learning Rate).
- Dropout Rate
- Selected Features

This algorithm considers three key variables - alpha (α), beta (β) and delta (δ) come from... The rest of the scenarios for these wolves show possible solutions, and the hunting process is compared as an optimization process.

- Alpha wolves (α) lead the hunt, representing the best solutions.
- Beta (β) and Delta (δ) wolves help Alpha and explore the search area properly.
- Omega wolves (ω), representing the rest of the population, follow these leads and converge to the optimal solution

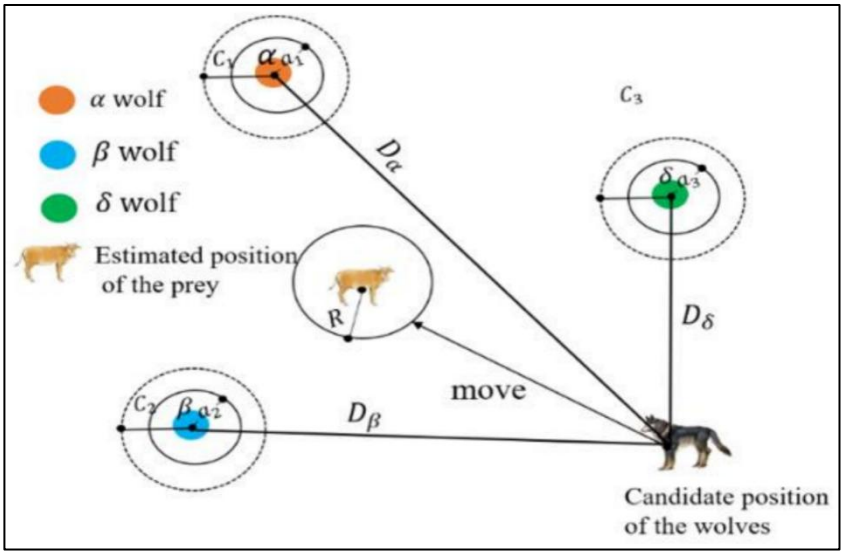

Fig 6.5: Schematic Representation of the Gray Wolf Optimizer's Hunting Behavior

The new position of a wolf is calculated as the average of three positions determined by the influence of the alpha, beta, and delta wolves. Fig. 6.5, provides how calculations will be done in the positions

$$\mathbf{Z(t+1)} = \frac{Z1+Z2+Z3}{3} \quad (1)$$

Each of the positions Z_1, Z_2, and Z_3 is calculated using the following equations

$$\mathbf{Z_1 = \alpha(t) - P_1 \cdot A_\alpha} \quad (2)$$

$$\mathbf{Z_2 = \beta(t) - P_2 \cdot A_\beta} \quad (3)$$

$$\mathbf{Z_3 = \delta(t) - P_3 \cdot A_\delta} \quad (4)$$

The distance between a wolf's position and the prey (best solution) is calculated as

$$\mathbf{A_\alpha = |Q_1 \cdot \alpha(t) - Z(t)|} \quad (5)$$

$$\mathbf{A_\beta = |Q_2 \cdot \beta(t) - Z(t)|} \quad (6)$$

$$\mathbf{A_\delta = |Q_3 \cdot \delta(t) - Z(t)|} \quad (7)$$

The coefficients P and Q are calculated using the following formulas

$$\mathbf{P = 2 \cdot b \cdot s_1 - b} \quad (8)$$

$$\mathbf{Q = 2 \cdot s_2} \quad (9)$$

Where s_1 and s_2 are random numbers in the range [0, 1].

The parameter $\boldsymbol{\alpha}$ decreases linearly throughout iterations to balance exploration and exploitation

$$\mathbf{a = 2 - \frac{2 \cdot t}{T}} \quad (10)$$

t is the current iteration.

T is the maximum number of iterations.

These equations are fundamental to the operation of the GWO algorithm, guiding the wolves toward the optimal solution in the search space.

We ran GWO with 10 wolves (potential solutions) over 20 iterations. The optimization process updated the wolves' positions based on their distance from the best solution, ultimately converging to the optimal set of hyperparameters and selected features. Additionally, the algorithm selected approximately 10,316 features from the 25088 features, significantly reducing the dimensionality and improving the model's performance. Using GWO for hyperparameter optimization and feature selection effectively enhanced the CNN model's performance while reducing computational complexity. This approach can be extended to other high-dimensional datasets for various machine-learning tasks

6.6 Model Building:

OpCoNet (Optimized Convolutional Network), a highly specialized Deep Learning model and a user-defined model, was developed to classify the stages of Diabetic Retinopathy, an eye condition. Convolutional Neural Networks (CNNs) are used in conjunction with advanced optimization techniques by the model to enhance performance by classifying picture information. OpCoNet incorporates the Gray Wolf Optimizer (GWO) algorithm, a nature-inspired optimization technique that mimics Gray wolves' leadership structure and hunting tactics in the wild. GWO is used to optimize the network's hyperparameters and extract the most relevant features from the high-dimensional picture data to ensure the model's accuracy and efficiency. OpCoNet is meant to handle the complexity and unpredictability of medical imaging, where minute differences in image properties may be crucial for an accurate diagnosis. The model starts with an input layer that processes a set of carefully selected features from the original dataset based on their relevance. The first significant step involves a thick layer in which each neuron takes input and performs a weighted sum followed by a non-linear activation function, allowing the model to learn complicated correlations within the data. To prevent overfitting approaches such as L2 regularization are used, which helps the model avoid becoming overly specialized for the training dataset.

Next, the model uses batch normalization to stabilize and accelerate training by normalizing the dense layer's output.

Table 6.1 The hyperparameters for the model construction.

Hyper Parameters	**Value**
Neurons_Layer1	181
Learning Rate	0.0004572052659916009
Neurons_Layer2	150
Dropout_Rate	0.1
L2 Regularization	0.001
Loss	Categorical Cross Entropy

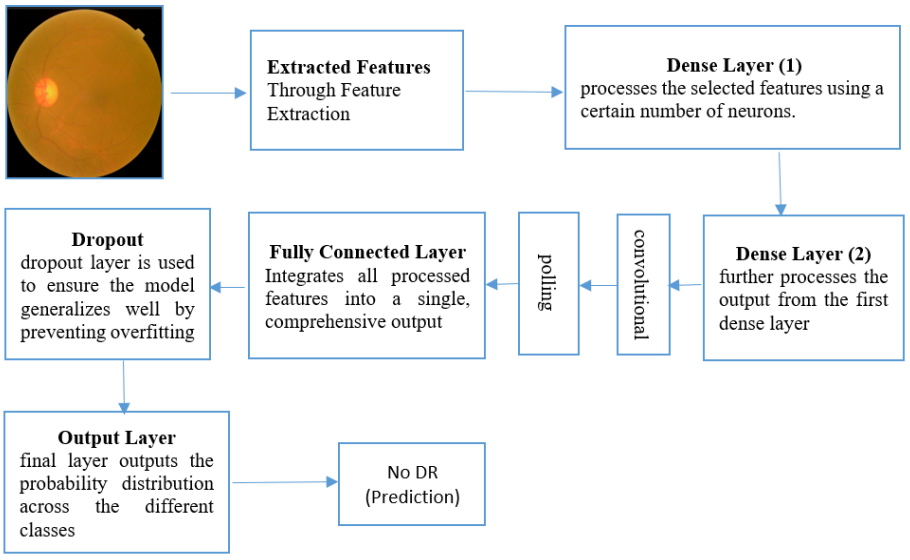

Fig 6.6 The Model Architecture

A dropout layer is also present, randomly deactivating some neurons during training to ensure that the model does not grow unduly reliant on any specific neurons, preventing overfitting. This same process is repeated in a second dense layer, though with a different number of neurons, again followed by batch normalization and dropout as shown in Fig. 6.6. After these dense layers, the model introduces a Convolutional layer that applies filters to detect intricate patterns within the data.

This is followed by a pooling layer, which reduces the size of the data, retaining the most important information while making the model more efficient. Finally, the processed information is passed through a fully connected layer, leading to the output layer. This final layer uses a SoftMax activation function to produce a probability distribution, allowing the model to predict the most likely category for each input. Several hyperparameters in Table 6.1, are carefully controlled throughout the process, including the number of neurons in each dense layer, learning rate, dropout rate, and L2 regularization strength. The model is trained with an optimizer that adjusts the weights based on the error, which is calculated using categorical cross-entropy loss. Early stopping is used to prevent the model from overtraining, which halts training when no further improvement is observed. This strategy produces a well-rounded model that can accurately identify the incoming data.

6.7 Classification:

The proposed classification model is a fully connected neural network designed to predict the severity of Diabetic Retinopathy based on extracted features. The architecture in Fig. 6.7, consists of multiple dense layers, batch normalization, and dropout layers, which work together to improve learning efficiency and prevent overfitting.

The model begins with an initial dense layer that learns high-level feature representations, followed by a batch normalization layer to stabilize and accelerate training. A dropout layer is included to reduce overfitting by randomly deactivating neurons. Subsequent dense layers refine the learned features, with each layer followed by batch normalization and dropout for better generalization. The final output layer consists of multiple neurons corresponding to different severity levels of Diabetic Retinopathy, using a SoftMax activation function to predict the most probable class. This structured approach ensures effective feature extraction, stable learning, and improved classification accuracy, making the model suitable for automated Diabetic Retinopathy detection. The proposed methodology provides an efficient way to detect the DR with the help of Gray Wolf Optimization. This way promotes faster results than other systems in terms of accuracy of 99.31% and less time consumption for model training.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 181)	1,867,377
batch_normalization (BatchNormalization)	(None, 181)	724
dropout (Dropout)	(None, 181)	0
dense_1 (Dense)	(None, 150)	27,300
batch_normalization_1 (BatchNormalization)	(None, 150)	600
dropout_1 (Dropout)	(None, 150)	0
dense_2 (Dense)	(None, 5)	755

Total params: 1,896,758 (7.24 MB)
Trainable params: 1,896,094 (7.23 MB)
Non-trainable params: 662 (2.59 KB)
Optimizer params: 2 (12.00 B)
Number of selected features: 10316

Fig 6.7 OpCoNet with GWO Selected Features Model Architecture

To assess the performance of our classification model, we employ various evaluation metrics that provide insights into its effectiveness in identifying Diabetic Retinopathy. The key metrics used include accuracy, precision, recall, F1 score, time complexity, and the number of features selected.

6.8 Evaluation Metrics

- **Accuracy:**

Accuracy measures the proportion of correctly classified instances out of the total instances. It is a fundamental metric for assessing model performance.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

where TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

- **Precision**:

Precision quantifies how many of the predicted positive instances are actually positive. It is crucial when false positives need to be minimized.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall:**

  Recall measures the model's ability to identify actual positive cases. A high recall indicates that the model captures the most relevant instances.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- **F1 Score**:

The F1 Score is the harmonic mean of precision and recall, providing a balanced measure when both are important.

$$\text{F1 Score} = \frac{Precision * Recall}{Precision + Recall}$$

- **Time Complexity**:

Time complexity refers to the computational cost of running the model in terms of processing time. It depends on factors such as dataset size, number of features, and algorithm efficiency.

- **Number of Features Selected**:

The extracted features from the VGG16 model are used for classification. Reducing redundant features while preserving important patterns helps in improving classification speed and reducing overfitting, leading to better generalization.

7. DESIGN

The proposed framework in Fig. 7.1, for Diabetic Retinopathy (DR) classification begins with input retinal images from the "Diabetic Retinopathy Level Detection" dataset. These high-resolution images undergo preprocessing to ensure uniformity and enhance model performance. The preprocessing includes resizing the images to 224x224 pixels, normalizing pixel values to a range of 0 to 1, and calculating class weights to address imbalances in the dataset. These steps prepare the images for feature extraction and classification, ensuring consistency across the training and validation phases.

The pre-trained VGG16 model is utilized for feature extraction, leveraging its depth and proven capability in image classification tasks. VGG16 generates high-dimensional feature vectors (25,088 features per image) from the preprocessed images. The Gray Wolf Optimizer (GWO) is applied to optimize the feature set, reducing the dimensionality to 10,316 features. This optimization step significantly lowers computational costs and ensures that only the most relevant features contribute to the classification process, improving the system's efficiency and accuracy.

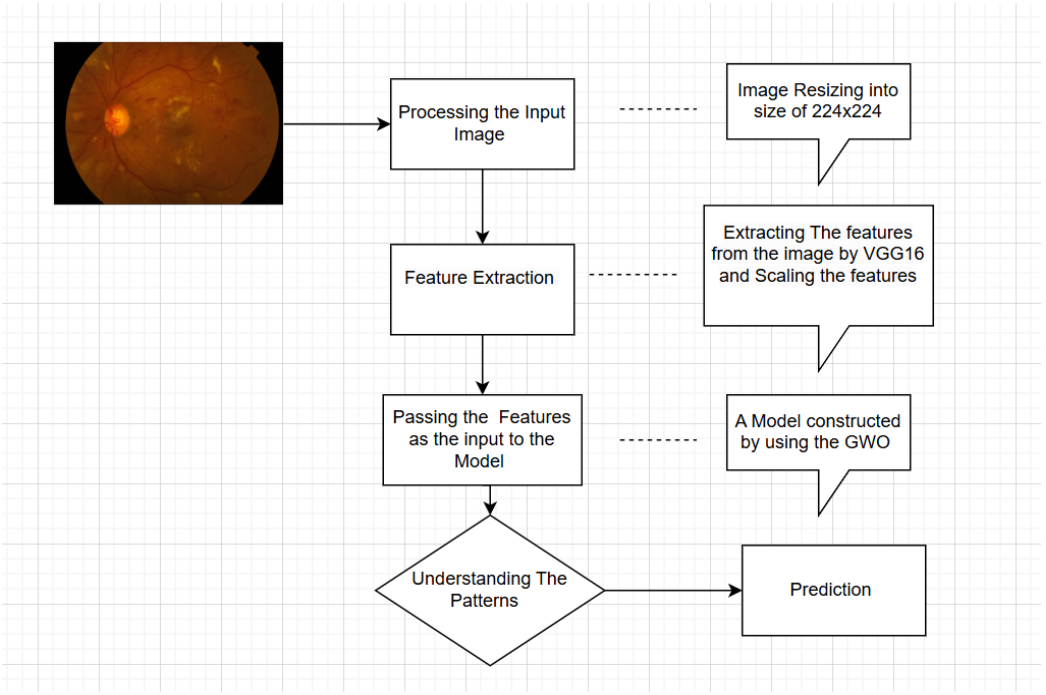

Fig 7.1 Design Overview

The optimized features are passed to the Optimized Convolutional Neural Network (OpCoNet) for classification. The CNN architecture includes fine-tuned hyperparameters, such as the number of neurons, learning rate, and dropout rate, which were optimized using GWO. Batch normalization and dropout layers are integrated to enhance generalization and reduce overfitting. The final SoftMax layer categorizes the images into five DR severity levels, ranging from no DR (Level 0) to severe proliferative DR (Level 4). This structured pipeline ensures the model's robustness, computational efficiency, and applicability in real-world clinical settings.

8. IMPLEMENTATION

Feature Extraction

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.utils import to_categorical
# Paths to your training and validation directories
train_dir = '/content/drive/MyDrive/dataset/preprocessed dataset/preprocessed dataset/training'
val_dir = '/content/drive/MyDrive/dataset/preprocessed dataset/preprocessed dataset/testing'
datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='sparse',
    shuffle=False )
val_generator = datagen.flow_from_directory(
    val_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='sparse',
    shuffle=False )
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
def extract_features(generator, base_model):
    features_list = []
    labels_list = []
    for inputs_batch, labels_batch in generator:
        features_batch = base_model.predict(inputs_batch)
```

```
        features_list.append(features_batch)
        labels_list.append(labels_batch)
        if len(features_list) * generator.batch_size >= generator.samples:
            break
    features = np.concatenate(features_list, axis=0)
    labels = np.concatenate(labels_list, axis=0)
    return features, labels
train_features, train_labels = extract_features(train_generator, base_model)
```

Optimization Algorithm (GWO)

```
def cnn_objective_function(params):
    # Extract hyperparameters and feature selection mask
    num_neurons1 = int(params[0])
    num_neurons2 = int(params[1])
    learning_rate = params[2]
    dropout_rate = params[3]
    feature_selection_mask = params[4:]
    # Ensure feature selection mask is binary
    feature_selection_mask = np.where(feature_selection_mask >= 0.5, 1, 0)
    # Select features based on mask
    selected_features = np.where(feature_selection_mask == 1)[0]
    if len(selected_features) == 0:
        return float('inf')  # Penalize solutions with no selected features
    X_train_selected = X_train_scaled[:, selected_features]
    X_val_selected = X_val_scaled[:, selected_features]
    # Build and compile the model
    model = Sequential([
        Dense(num_neurons1, activation='relu', input_shape=(X_train_selected.shape[1],)),
        Dropout(dropout_rate),
        Dense(num_neurons2, activation='relu'),
        Dropout(dropout_rate),
        Dense(num_classes, activation='softmax')  # Output layer with num_classes neurons ])
    model.compile(optimizer=Adam(learning_rate=learning_rate),
                  loss=CategoricalCrossentropy(),
                  metrics=['accuracy'])
    # Train the model
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=3,
restore_best_weights=True)
 model.fit(X_train_selected, train_labels, epochs=5, batch_size=32, verbose=0,
validation_data=(X_val_selected, val_labels),
callbacks=[early_stopping],class_weight=class_weights_dict)
    # Evaluate the model
    val_loss, val_accuracy = model.evaluate(X_val_selected, val_labels, verbose=0)
    return -val_accuracy  # Return negative accuracy
def gwo_optimize(obj_function, num_wolves, num_iterations, dim, lower_bound,
upper_bound):
    alpha_pos = np.zeros(dim)
    alpha_score = float('inf')
    beta_pos = np.zeros(dim)
    beta_score = float('inf')
    delta_pos = np.zeros(dim)
    delta_score = float('inf')
    wolves_positions = np.random.uniform(lower_bound, upper_bound, (num_wolves, dim))
    for iteration in range(num_iterations):
        for i in range(num_wolves):
            fitness = obj_function(wolves_positions[i])
            if fitness < alpha_score:
                alpha_score = fitness
                alpha_pos = wolves_positions[i].copy()
            elif fitness < beta_score:
                beta_score = fitness
                beta_pos = wolves_positions[i].copy()
            elif fitness < delta_score:
                delta_score = fitness
                delta_pos = wolves_positions[i].copy()
        a = 2 - iteration * (2 / num_iterations)
        for i in range(num_wolves):
            for j in range(dim):
                r1, r2 = np.random.rand(2)
                A1 = 2 * a * r1 - a
                C1 = 2 * r2
                D_alpha = abs(C1 * alpha_pos[j] - wolves_positions[i, j])
                X1 = alpha_pos[j] - A1 * D_alpha
```

```
            r1, r2 = np.random.rand(2)
            A2 = 2 * a * r1 - a
            C2 = 2 * r2
            D_beta = abs(C2 * beta_pos[j] - wolves_positions[i, j])
            X2 = beta_pos[j] - A2 * D_beta
            r1, r2 = np.random.rand(2)
            A3 = 2 * a * r1 - a
            C3 = 2 * r2
            D_delta = abs(C3 * delta_pos[j] - wolves_positions[i, j])
            X3 = delta_pos[j] - A3 * D_delta
            wolves_positions[i, j] = (X1 + X2 + X3) / 3
            wolves_positions[i, j] = np.clip(wolves_positions[i, j], lower_bound[j],
upper_bound[j])
      print(f"Iteration {iteration+1}/{num_iterations}, Alpha Score: {-alpha_score}")
   return alpha_pos, alpha_score
num_wolves = 10
num_iterations = 20
dim = 4 + 25088  # 4 for hyperparameters and 25,088 for feature selection
lower_bound = [16, 16, 0.0001, 0.1] + [0] * 25088
upper_bound = [256, 256, 0.01, 0.5] + [1] * 25088
best_params, best_score = gwo_optimize(cnn_objective_function, num_wolves,
num_iterations, dim, lower_bound, upper_bound)
# Extract the best hyperparameters
num_neurons1 = int(best_params[0])
num_neurons2 = int(best_params[1])
learning_rate = best_params[2]
dropout_rate = best_params[3]
feature_selection_mask = best_params[4:]
# Get the selected features
selected_features = np.where(feature_selection_mask >= 0.5)[0]
# Save the selected features indices
np.save('/content/drive/MyDrive/selected_feature_indices_byGWO.npy', selected_features)
```

Model Building Code

```
#Model (OpCoNet) By using the GWO Feature Selection
# Input is Extracted Features and the output is Model
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.preprocessing import StandardScaler
import pickle
from tensorflow.keras.callbacks import EarlyStopping
# Assuming train_features, val_features, train_labels, and val_labels are already
defined
# And that you've already selected the features as per the selected indices
# Parameters from the GWO optimization
neurons_layer1 = 212
neurons_layer2 = 199
learning_rate = 0.0005706062976624917
dropout_rate = 0.16427840128721727
num_classes = 5
epochs = 25
batch_size = 32
# Feature Scaling
scaler = StandardScaler()
# # Load the selected feature indices
selected_features =
np.load('/content/drive/MyDrive/selected_feature_indices_byGWO.npy')
# Define function for feature selection
def select_features(X, selected_features):
    return X[:, selected_features]
# Select features for training and validation data
X_train_selected = select_features(X_train_scaled, selected_features)
X_val_selected = select_features(X_val_scaled, selected_features)
# Build the model using the optimized parameters
# Example of adding L2 regularization and Batch Normalization
from tensorflow.keras import regularizers
model = models.Sequential([
    layers.Dense(neurons_layer1, activation='relu',
input_shape=(len(selected_features),),kernel_regularizer=regularizers.l2(0.001)),
```

```
    layers.BatchNormalization(),
    layers.Dropout(dropout_rate),
    layers.Dense(neurons_layer2, activation='relu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.BatchNormalization(),
    layers.Dropout(dropout_rate),
    layers.Dense(num_classes, activation='softmax')
])
# You can keep the rest of the training and evaluation code the same.
# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# Early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=3,
restore_best_weights=True)
# Train the model on the selected features
history = model.fit(
    X_train_selected, train_labels,
    epochs=epochs,
    batch_size=batch_size,
    validation_data=(X_val_selected, val_labels),
    callbacks=[early_stopping],class_weight=class_weights_dict )
# Evaluate the model on validation set
val_loss, val_accuracy = model.evaluate(X_val_selected, val_labels, verbose=2)
print(f"Validation accuracy: {val_accuracy}")
# Save the model
model.save('/content/drive/MyDrive/optimized_feature_extracted_GWO_Model.h5')
# Save the model's architecture and weights separately
model_structure = model.to_json()
model_weights = model.get_weights()
# Serialize the architecture and weights using pickle
with open('/content/drive/MyDrive/model_structure.pkl', 'wb') as file:
```

```
    pickle.dump(model_structure, file)
with open('/content/drive/MyDrive/model_weights.pkl', 'wb') as file:
    pickle.dump(model_weights, file)
```

Flask Front End Connection Code

```
import joblib
import cv2
from flask import Flask, request, render_template
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input, VGG16
from tensorflow.keras.models import load_model, Model
app = Flask(__name__)
# Load necessary models and scalers
try:
    selected_feature_indices = np.load('selected_feature_indices_2.npy')
    scaler = joblib.load('standard_scaler.pkl')
    model = load_model('optimized_extracted_model_2.h5')
    model.compile()
except Exception as e:
    print(f"Error loading model or scaler: {e}")
    selected_feature_indices = None
    scaler = None
    model = None
# Load VGG16 model for feature extraction
vgg16_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
feature_extractor = Model(inputs=vgg16_model.input, outputs=vgg16_model.output)
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
def allowed_file(filename):
    """Check if the uploaded file has a valid image extension."""
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
def is_retina_shape(img_path):
    try:
```

```
    # Read the image
    img = cv2.imread(img_path)
    if img is None:
        print("Error: Unable to read image.")
        return False
    # Convert to HSV and check color properties
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower_orange = np.array([5, 50, 50])  # Lower bound of orange-red hues
    upper_orange = np.array([30, 255, 255])  # Upper bound
    mask = cv2.inRange(hsv, lower_orange, upper_orange)
    orange_ratio = np.sum(mask > 0) / (img.shape[0] * img.shape[1])
    if orange_ratio < 0.05:  # Less than 5% orange pixels → Not a retina image
        print("Image rejected: Not enough orange color.")
        return False
    # Convert to Grayscale and apply CLAHE (Contrast Limited Adaptive
Histogram Equalization)
    Gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    Gray = clahe.apply(Gray)
    # Apply Gaussian Blur
    blurred = cv2.GaussianBlur(Gray, (5, 5), 0)
    # Detect circles using Hough Transform
    circles = cv2.HoughCircles(blurred, cv2.HOUGH_GRADIENT, 1, minDist=30,
                               param1=50, param2=30, minRadius=50, maxRadius=300)
    if circles is not None:
        print("✅ Retina image detected via HoughCircles.")
        return True
          edges = cv2.Canny(Gray, 50, 150)
    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        perimeter = cv2.arcLength(contour, True)
        area = cv2.contourArea(contour)
```

```
            if area > 500 and 0.7 < (4 * np.pi * area) / (perimeter ** 2) < 1.3:
                print("✅ Retina image detected via contour-based analysis.")
                return True
        print("❌ Image rejected: No retina-like circular structures detected.")
        return False
    except Exception as e:
        print(f"Error in retina validation: {e}")
        return False
def extract_features(img_path):
    try:
        img = image.load_img(img_path, target_size=(224, 224))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = preprocess_input(img_array)
        features = feature_extractor.predict(img_array)
        features_flat = features.reshape((features.shape[0], -1))
        features_scaled = scaler.transform(features_flat)
        selected_features = features_scaled[:, selected_feature_indices]
        return selected_features
    except Exception as e:
        print(f"Feature extraction error: {e}")
        return None
def get_eye_health_tips(pred_class):
    tips = {
        0: "Your eyes are healthy! Maintain a balanced diet and regular checkups.",
        1: "Mild DR detected. Consider an eye exam and monitor your blood sugar
levels.",
        2: "Moderate DR detected. Schedule a consultation with an eye specialist.",
        3: "Severe DR detected! Immediate medical intervention is recommended.",
        4: "Proliferative DR detected! Seek urgent treatment to prevent vision loss."
    }
    return tips.get(pred_class, "No specific advice available.")
@app.route("/")
```

```
def home():
    return render_template("h.html")
@app.route("/predict", methods=["GET", "POST"])
def predict():
    if request.method == "POST":
        if "image" not in request.files or request.files["image"].filename == "":
            return render_template("index.html", prediction="No Image Selected",
file_name=None)
        file = request.files["image"]
        if not allowed_file(file.filename):
            return render_template("index.html", prediction="Invalid File Format!",
file_name=None)
        file_path = os.path.join("uploads", file.filename)
        os.makedirs("uploads", exist_ok=True)
        file.save(file_path)
        # Validate retina shape
        if not is_retina_shape(file_path):
            os.remove(file_path)
            return render_template("index.html", prediction="Invalid Image! No retina
shape detected.", file_name=file.filename)
        # Feature Extraction & Prediction
        try:
            features = extract_features(file_path)
            if features is None:
                os.remove(file_path)
                return render_template("index.html", prediction="Feature extraction
failed!", file_name=file.filename)
            predictions = model.predict(features)
            predicted_class = np.argmax(predictions, axis=1)[0]
            confidence = np.max(predictions) * 100  # Convert to percentage
            health_tips = get_eye_health_tips(predicted_class)
            os.remove(file_path)
            return render_template("index.html", prediction=f"Predicted Class:
{predicted_class} (Confidence: {confidence:.2f}%)",
```

```
                health_tips=health_tips, file_name=file.filename)
        except Exception as e:
            return render_template("index.html", prediction=f"Error: {str(e)}",
file_name=file.filename)
    return render_template("index.html", prediction=None, file_name=None)
@app.route("/about")
def about():
    return render_template("about.html")
if __name__ == "__main__":
    app.run(debug=True)
```

Html Prediction Code

```
<body>
    <nav class="navbar">
        <div class="logo">OptiRetina AI</div>
        <ul class="nav-links">
            <li><a href="/">Home</a></li>
            <li><a href="/about">About DR</a></li>
        </ul>
    </nav>
    <div class="container">
        <h1>DR Detection</h1>
        <form method="POST" enctype="multipart/form-data">
            <div class="upload-box"
onclick="document.getElementById('image').click();">
                <label>
                    <i class="fas fa-cloud-upload-alt"></i>
                    Drag & Drop or Click to Upload
                </label>
                <input type="file" name="image" id="image" accept="image/*"
onchange="displayFileName()">
                <p id="file-name">No file selected</p>
            </div>
            <div class="image-preview" id="image-preview" style="display: none;">
```

```
            <h3>Selected Image:</h3>
            <img id="preview-img" src="" alt="Image Preview">
        </div>
        <input type="submit" value="Predict">
    </form>
    {% if prediction %}
        <p class="message">{{ prediction }}</p>
        <p class="confidence">{{ health_tips }}</p>
    {% endif %}
    {% if error %}
        <p class="error">{{ error }}</p>
    {% endif %}
</div>
<script>
    function displayFileName() {
        var input = document.getElementById("image");
        var fileName = input.files[0] ? input.files[0].name : "No file selected";
        document.getElementById("file-name").innerText = fileName;
        // Display Image Preview
        var previewContainer = document.getElementById("image-preview");
        var previewImage = document.getElementById("preview-img");
        var file = input.files[0];
        if (file) {
            var reader = new FileReader();
            reader.onload = function(e) {
                previewImage.src = e.target.result;
                previewContainer.style.display = "block"; };
            reader.readAsDataURL(file);
        } else {
            previewContainer.style.display = "none"; }
    }
</script>
</body>
```

9. RESULT ANALYSIS

1. OpCoNet With Images (GWO)

The best hyperparameters obtained using the Gray Wolf Optimization algorithm were used to build the Optimized Convolutional Neural Network (OpCoNet). This network accepts images as input and achieves an impressive accuracy of 98.23\% on a dataset containing 3,662 images

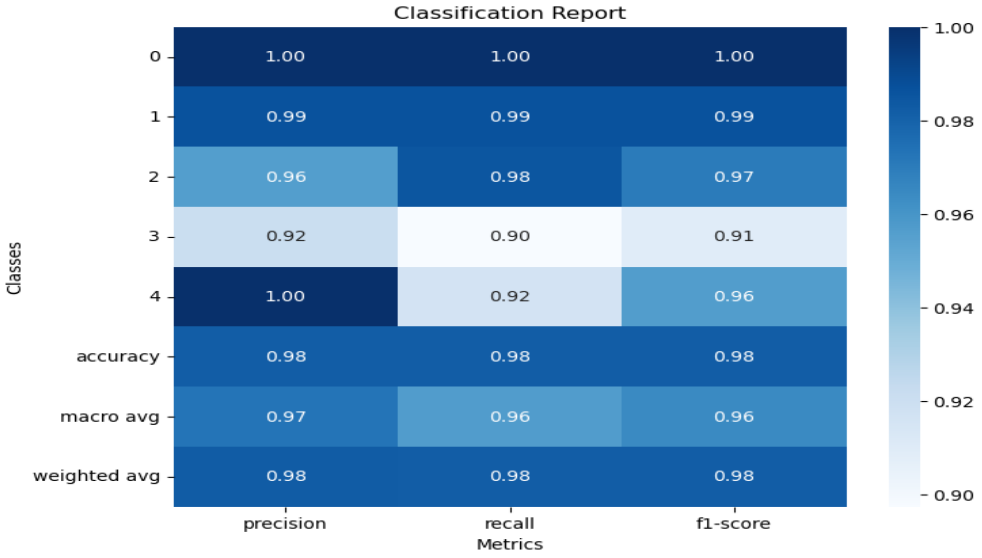

Fig 9.1 OpCoNet with Images (GWO) Classification Report

Fig. 9.1, presents the classification report for OpCoNet on the validation images, highlighting its performance metrics such as precision, recall, F1-score, and support for each class

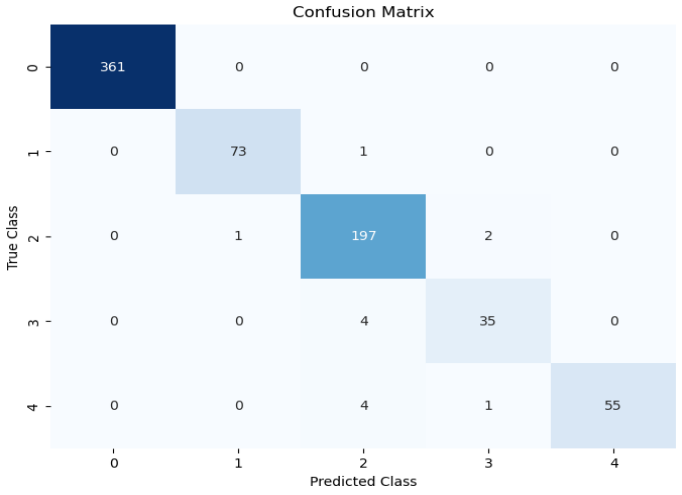

Fig 9.2 OpCoNet with Images (GWO) Confusion Matrix

Fig. 9.2, The Confusion Matrix provides a visual representation of the number of correct and incorrect predictions made by the model.

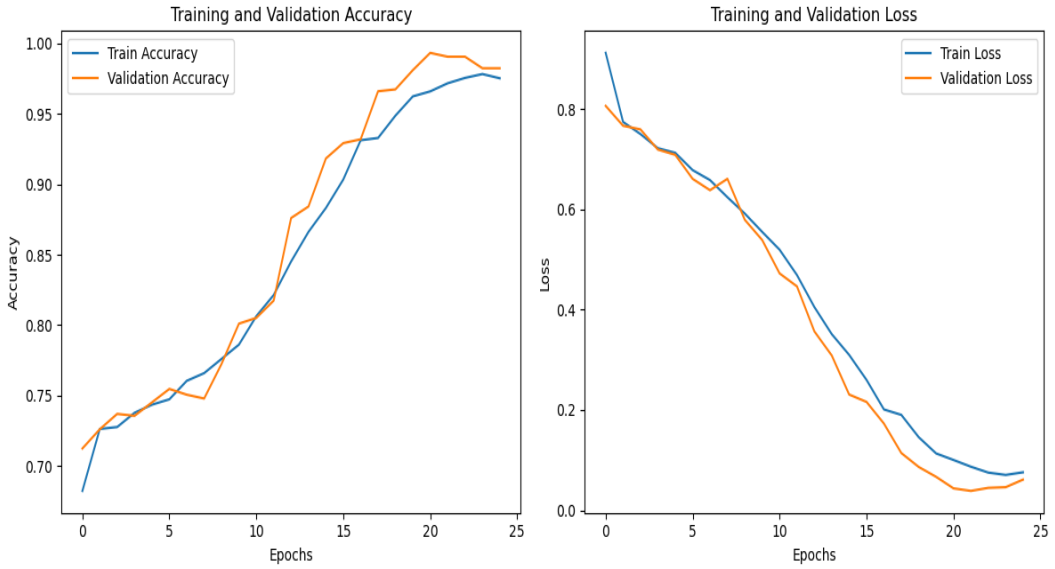

Fig 9.3 OpCoNet with Images (GWO) Model Performance Plots

The training and validation accuracy trend over epochs in Fig. 9.3, illustrates how well the model generalizes and learns over time, with the closing accuracy gap between training and validation indicating better performance and reduced overfitting. Similarly, the training and validation loss as a function of epochs demonstrates effective convergence and parameter optimization, as evidenced by the decreasing loss values. A steady decline in training and validation loss highlights the robust learning capability of the model.

2. OpCoNet With All Extracted Features (GWO)

The Gray Wolf Optimization algorithm determines the best hyperparameters to design the Optimized CNN (OpCoNet), which consumes the features extracted from each image in the dataset, approximately 25,088 features per image, with an accuracy of 99.18\% on the dataset of 3,662 images.

Fig. 9.4, depicts the classification report on the validation extracted features with metrics such as precision, recall, and F1-score. Fig. 9.5, The Confusion Matrix provides a visual representation of the number of correct and incorrect predictions made by the model.

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	361
1	0.99	1.00	0.99	74
2	0.98	0.99	0.99	200
3	1.00	0.90	0.95	39
4	0.98	0.98	0.98	60
accuracy			0.99	734
macro avg	0.99	0.98	0.98	734
weighted avg	0.99	0.99	0.99	734

Fig 9.4 OpCoNet with All Extracted Features (GWO) Classification Report

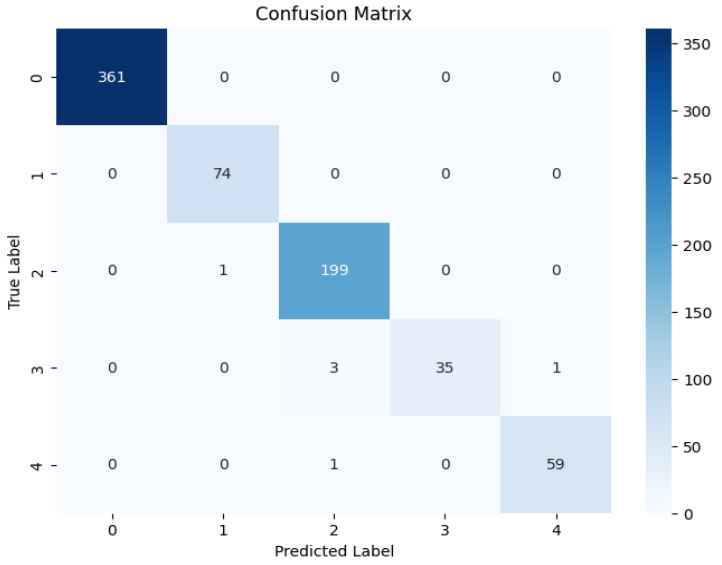

Fig 9.5 OpCoNet with All Extracted Features (GWO) Confusion matrix

The two subplots of Fig. 9.6, represent the performance of the model over 30 epochs. The first subplot illustrates the training and validation accuracy trend, where the orange line represents validation accuracy and the blue line represents training accuracy. In the early stages of training, the model performs very well; a reduced overfitting and an improved generalization are indicated by the closing gap between the two lines. The second subplot exhibits training and validation loss curves. The blue line indicates that it is the training loss whereas orange is the validation one. Steadily lowering the loss values indicates perfect convergence and optimization of the models for both.

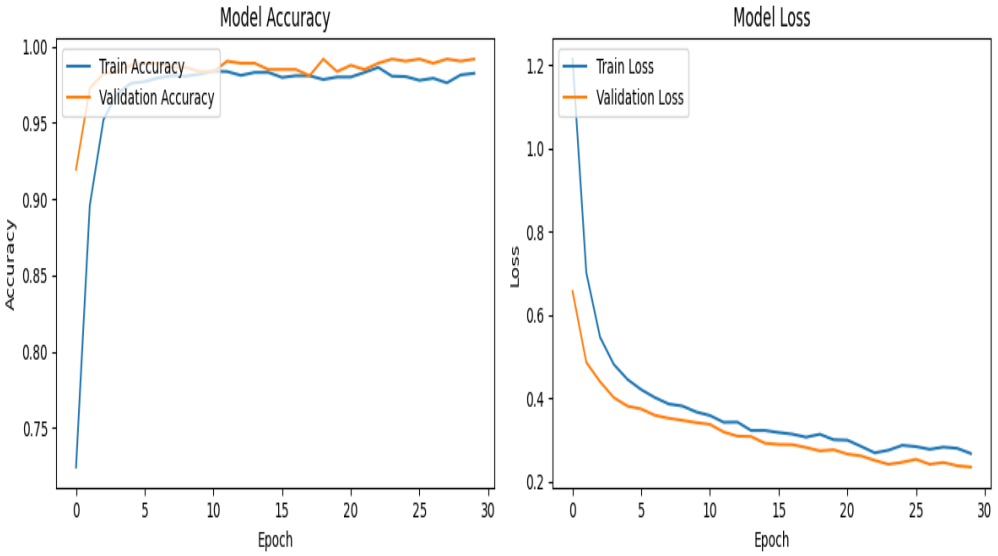

Fig 9.6 OpCoNet with All Extracted Features (GWO) Model Performance Plots

3. OpCoNet With ACO Selected Features

Ant Colony Optimization (ACO) operates similarly to Gray Wolf Optimization (GWO) but employs a different feature selection strategy. The Optimized CNN (OpCoNet) was built using the hyperparameters derived from ACO, selecting approximately 12,565 features from the initial 25,088 features per image. This optimized feature set enabled the model to achieve an impressive accuracy of 99.45% on a dataset of 3,662 images.

Classification Report:

	precision	recall	f1-score	support
0	1.000000	1.000000	1.000000	361.00000
1	0.986667	1.000000	0.993289	74.00000
2	0.994975	0.990000	0.992481	200.00000
3	0.950000	0.974359	0.962025	39.00000
4	1.000000	0.983333	0.991597	60.00000
accuracy	0.994550	0.994550	0.994550	0.99455
macro avg	0.986328	0.989538	0.987878	734.00000
weighted avg	0.994630	0.994550	0.994570	734.00000

Fig 9.7 OpCoNet with ACO Selected Features Classification Report

Fig. 9.7, presents the classification report of OpCoNet on the validation extracted features, showcasing metrics like precision, recall, and F1-score.

Fig. 9.8, illustrates the number of correct and incorrect predictions made by the model.

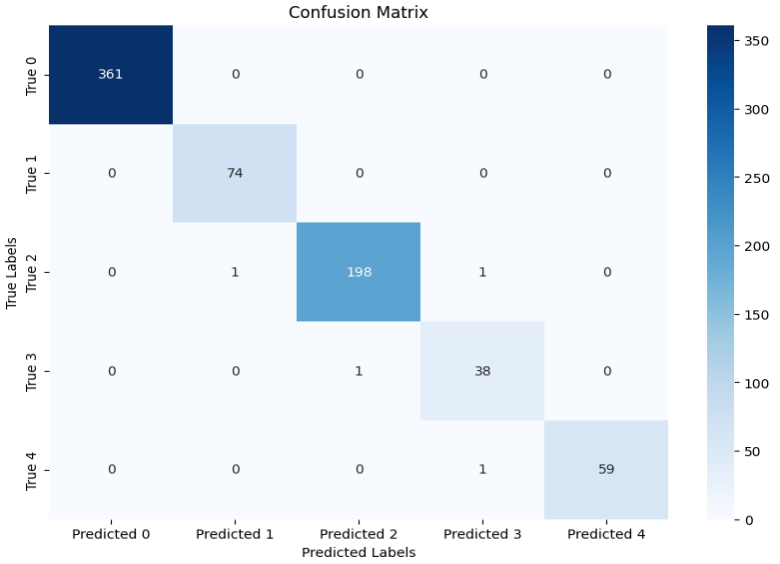

Fig 9.8 OpCoNet with ACO Selected Features Confusion Matrix

The Fig. 9.9, illustrates how the model was doing after 25 epochs in terms of accuracy and loss. The plot for accuracy indicates perfect generalization without overfitting since it displays that both training and validation accuracy surged very rapidly over the first few epochs and then stabilized very close to 1.0 with a small gap between the two curves. The hallmark of good model optimization, effective learning, and strong performance is the tight convergence of the training and validation loss curves showing a consistent drop in both.

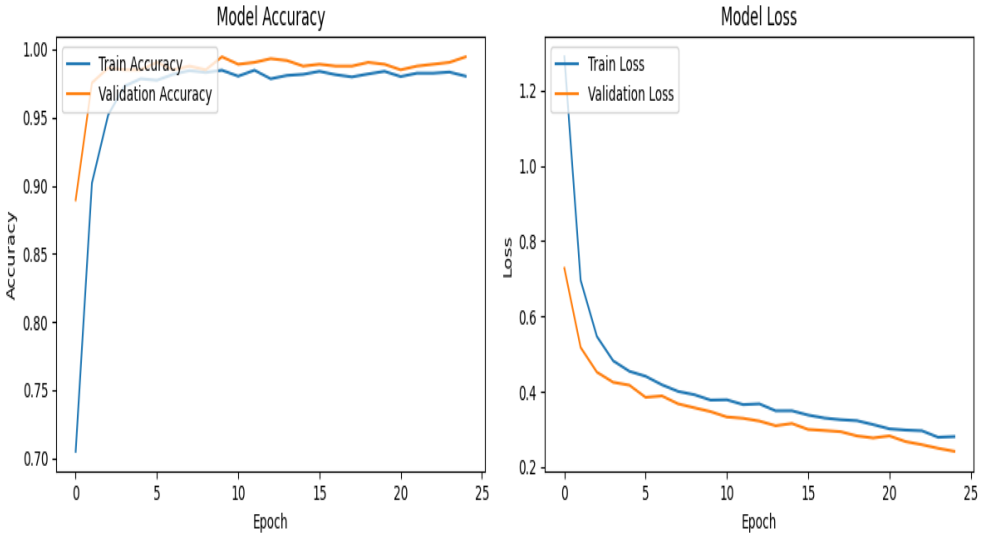

Fig 9.9 OpCoNet with ACO Selected Features Model Performance Plots

4. OpCoNet With GWO Selected Features

The Gray Wolf Optimization algorithm identified the best hyperparameters to build the Optimized CNN (OpCoNet), which accepts extracted features as input. Initially, each image had approximately 25,088 features, but the GWO reduced the feature set to 10,316 features. With this optimized feature set, the OpCoNet achieved an impressive accuracy of 99.31% on a dataset of 3,662 images.

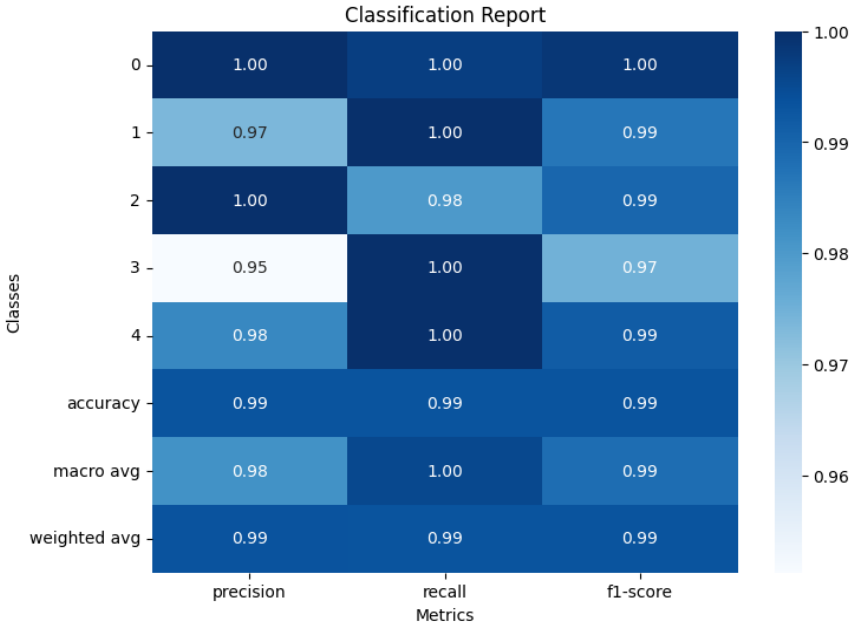

Fig 9.10 OpCoNet with GWO Selected Features Classification Report

Fig. 9.10, presents the classification report for OpCoNet on the validation extracted features, showcasing metrics like precision, recall, and F1-score. Fig. 9.11, illustrates the number of correct and incorrect predictions made by the model.

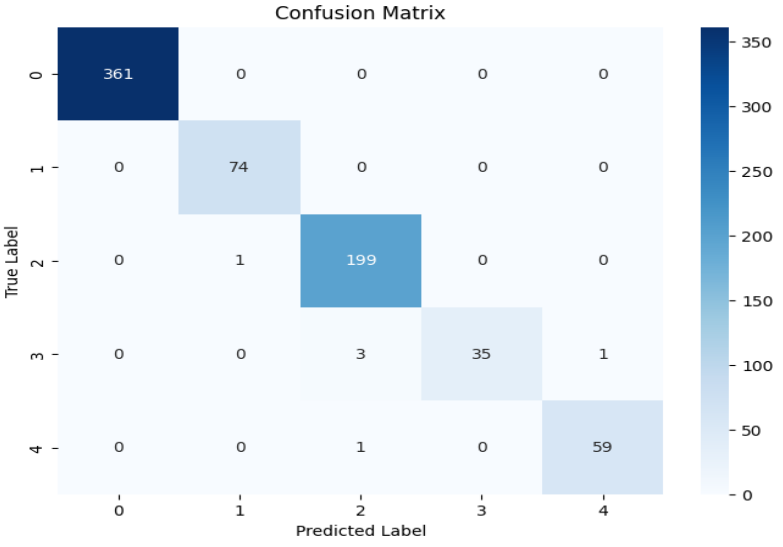

Fig 9.11 OpCoNet with GWO Selected Features Confusion Matrix

The model's performance over 25 epochs is depicted in Fig. 9.12, which shows strong generalization and efficient learning without any overfitting. Consistent performance on both training and validation datasets is indicated by the accuracy plot's training and validation accuracy curves, which both rise quickly during the first few epochs and stabilize near 1.0 with little space between them. The training and validation loss curves in the loss plot exhibit smooth convergence and a steady decline, indicating effective optimization and the lack of divergence. These patterns demonstrate how well-trained the model is and how well it generalizes to new data.

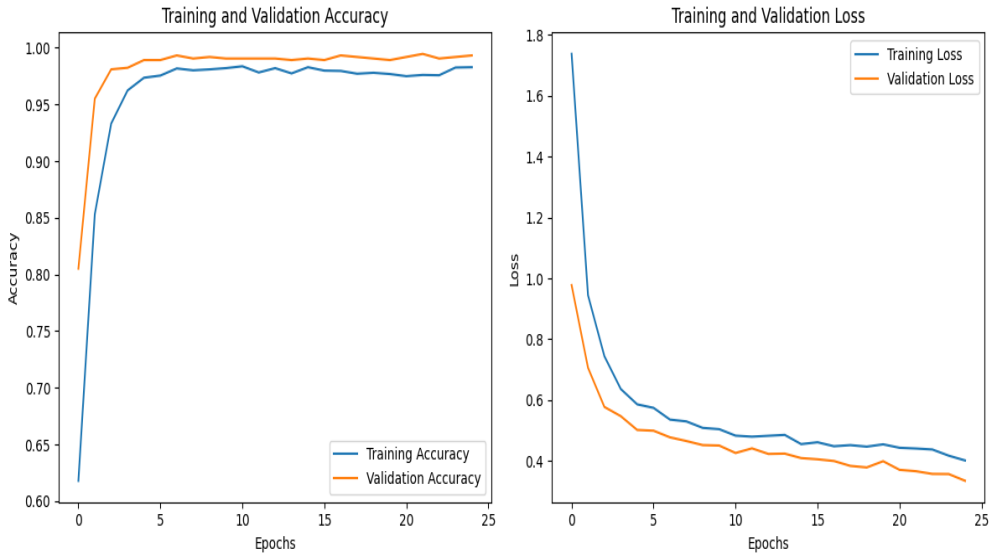

Fig 9.12 OpCoNet with GWO Selected Features Model Performance Plots

10. TEST CASES

Test Case 1: No DR

The retina in Fig. 10.1, appears normal with no visible abnormalities. There are no signs of microaneurysms, hemorrhages, or vascular changes**.**

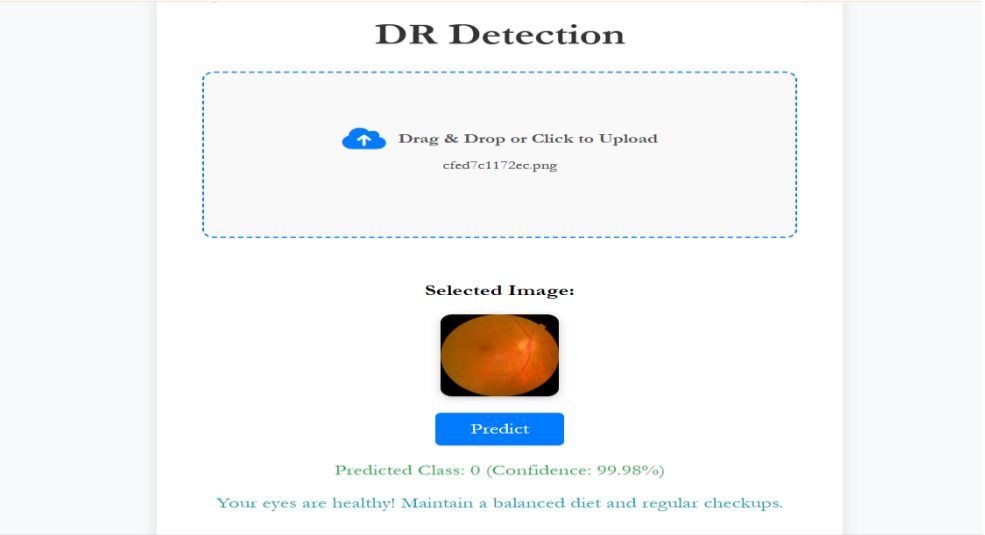

Fig 10.1 No DR Stage

Test Case 2: Mild DR

Small microaneurysms (tiny red dots) are present in Fig. 10.2, indicating early-stage Diabetic Retinopathy. There is minimal impact on vision at this stage.

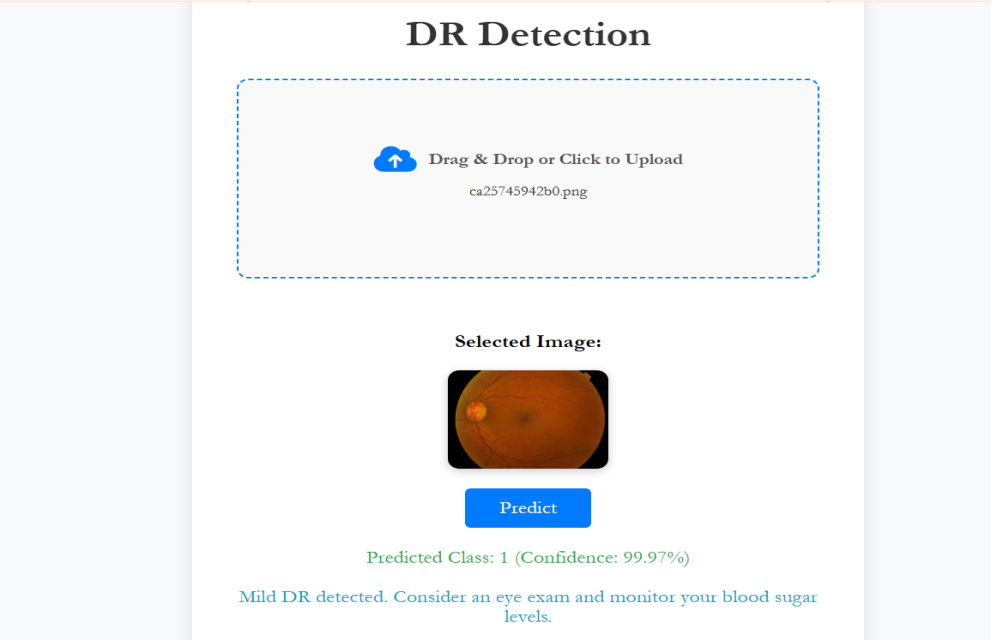

Fig 10.2 Mild DR Stage

Test Case 3: **Moderate DR**

An increased number of microaneurysms, hemorrhages, and mild vascular abnormalities are visible in Fig. 10.3. The risk of vision impairment begins to increase.

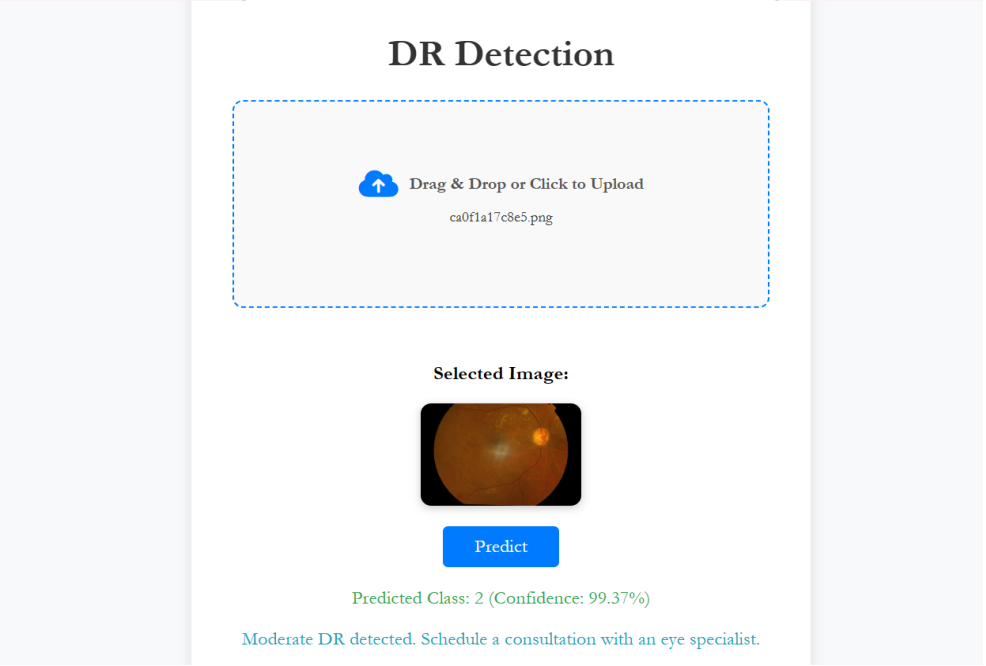

Fig 10.3 Moderate DR

Test Case 4: Serve DR

Multiple hemorrhages, venous beading, and significant retinal damage are observed in Fig. 10.4. The retina is at high risk of progressing to the proliferative stage.

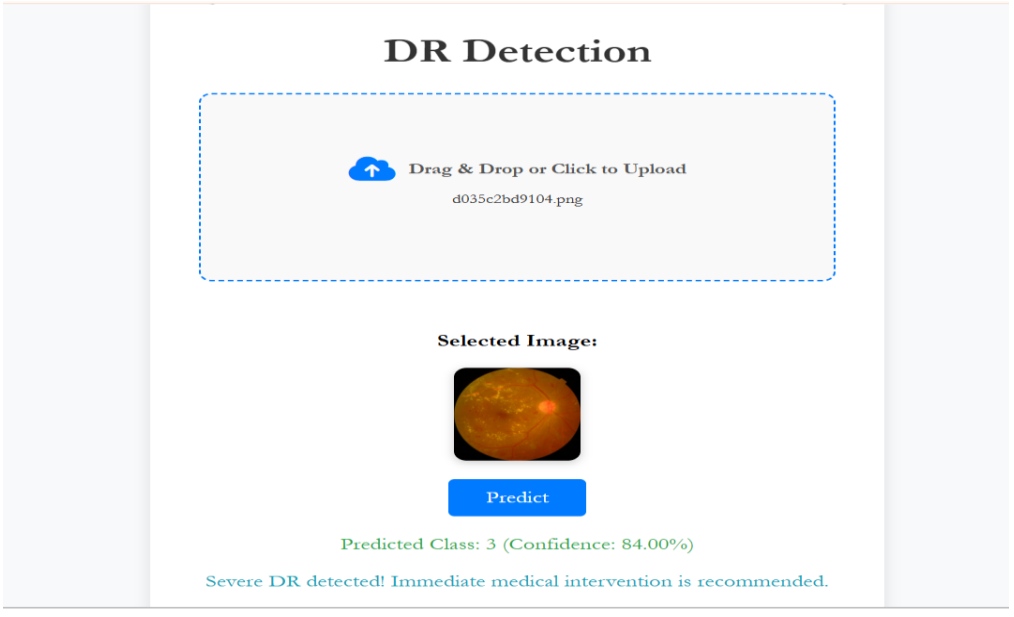

Fig 10.4 Serve DR Stage

Test Case 5: Proliferative DR

Abnormal new blood vessels (neovascularization) form, leading to potential retinal detachment in Fig. 10.5. This stage poses a severe risk of vision loss if untreated.

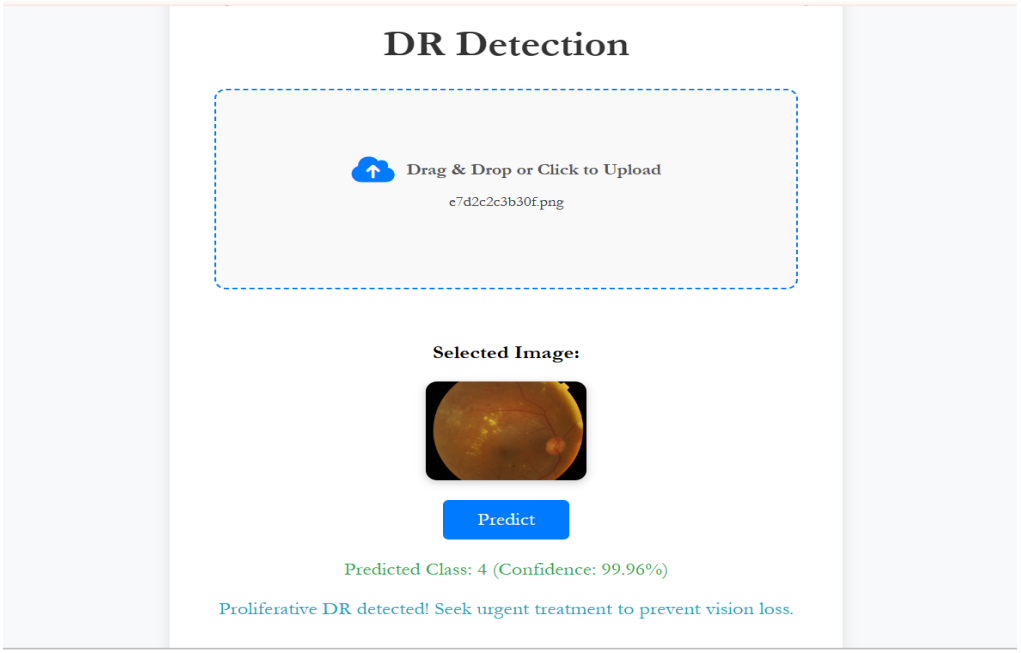

Fig 10.5 Proliferative DR

Test Case 6: Determining if the image is a retinal image.

When the user uploads a non-retinal image, the model detects it and provides an error message, asking the user to upload a proper retinal image**.**

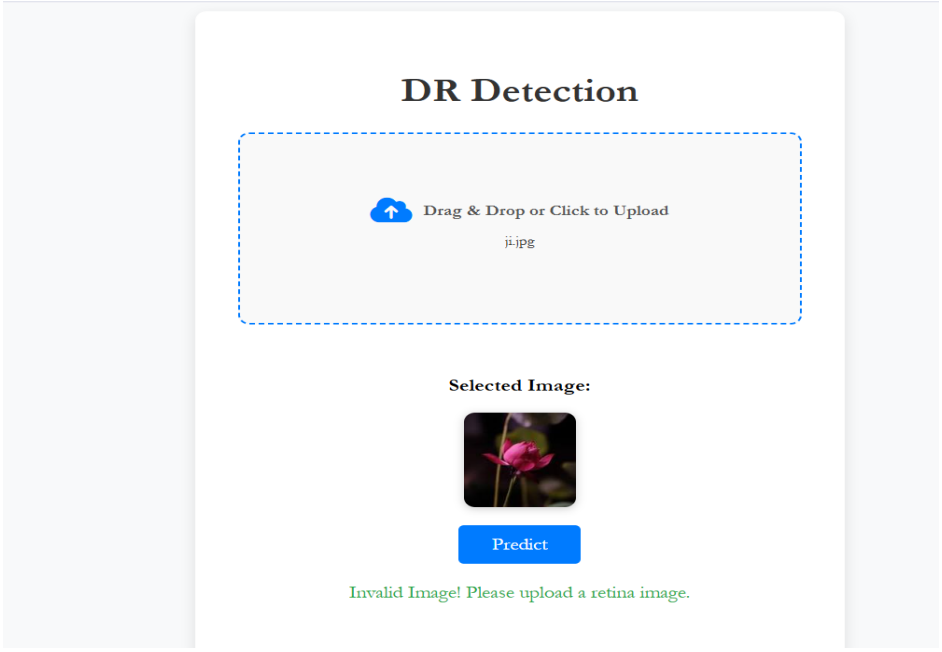

Fig 10.6 Validating whether the input image is retinal**.**

11. USER INTERFACE

The user interface of the Diabetic Retinopathy Detection System is designed for simplicity and ease of use. It includes a home screen (Fig 11.1) where users can upload retinal images for analysis. The system processes the image and classifies the severity of DR, displaying the corresponding stage of DR (Figs 11.2 – 11.5)

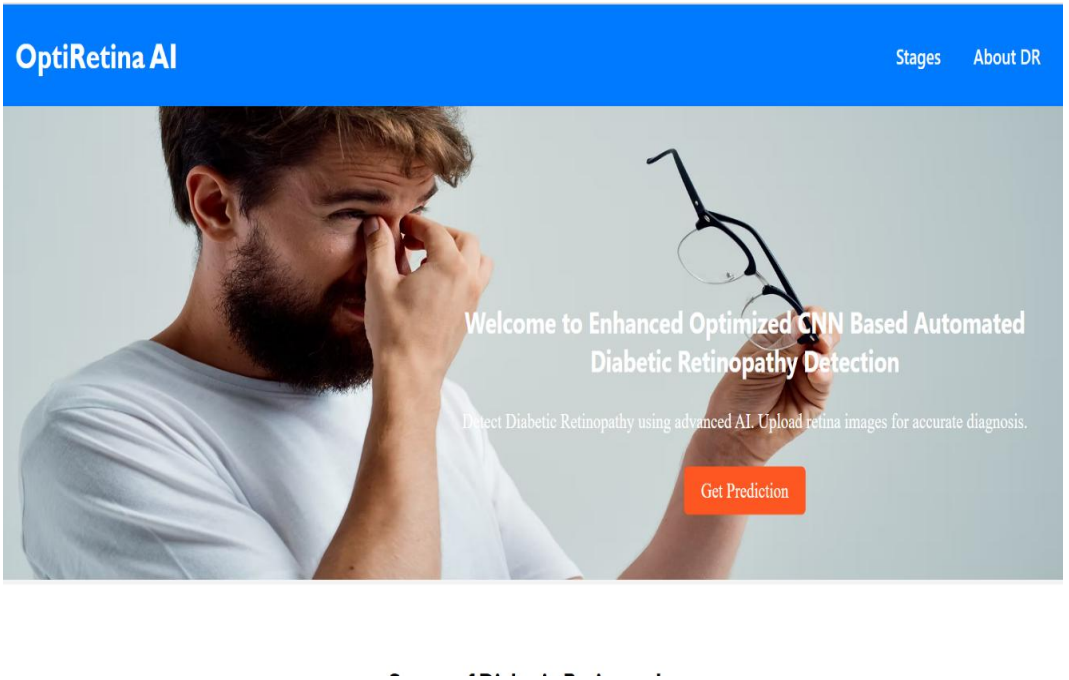

Fig 11.1 Home Screen

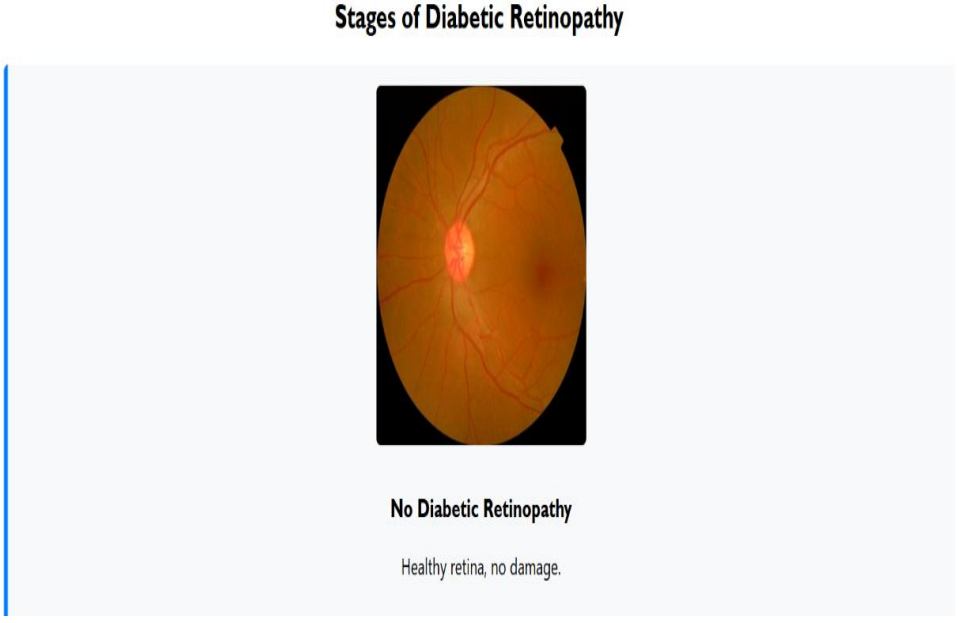

Fig 11.2 Stages of DR (Level 0)

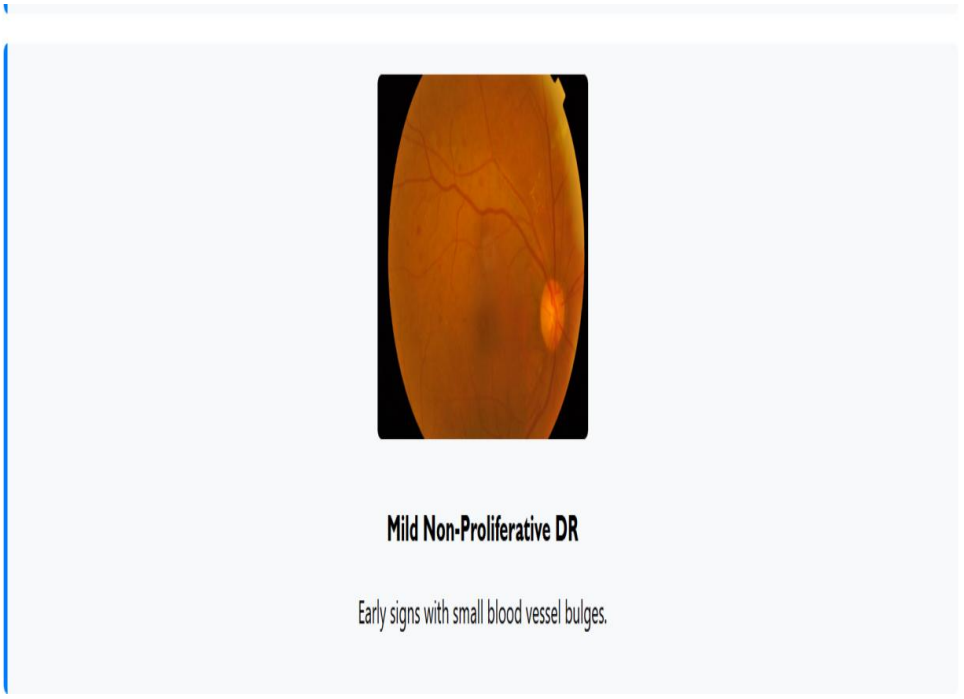

Fig 11.3 Stages of DR (Level-1)

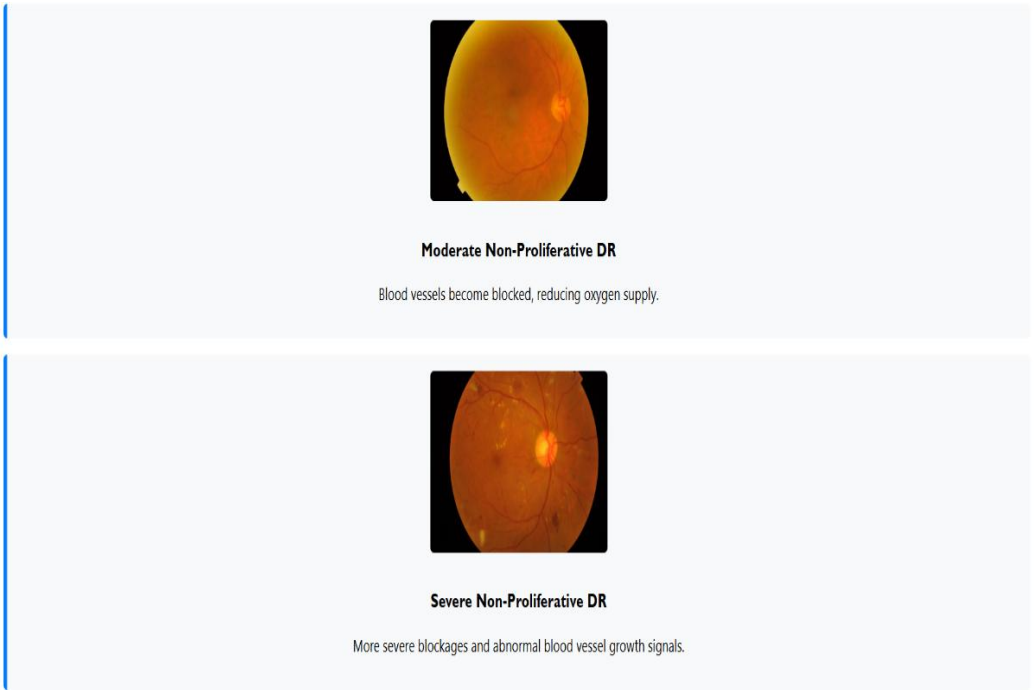

Fig 11.4 Stages of DR (Level-2 & Level -3)

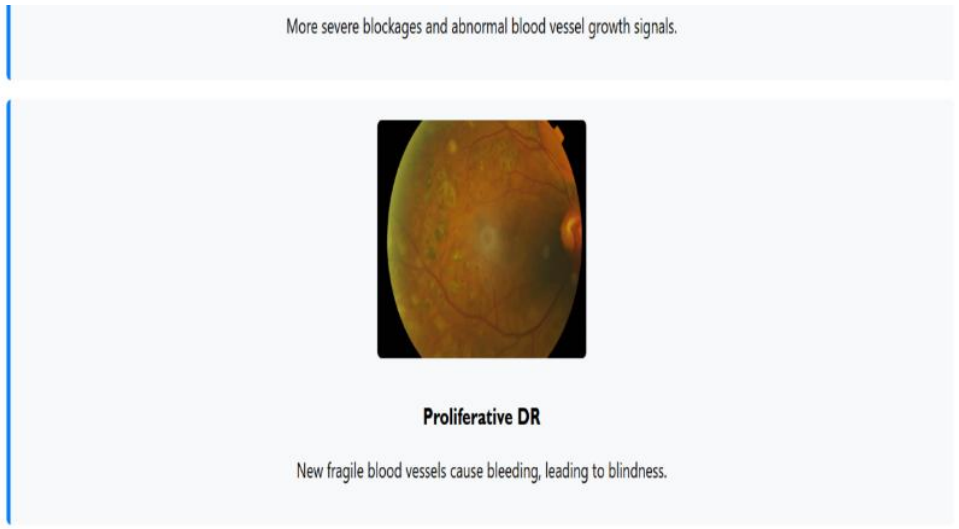

Fig 11.5 Stages of DR (Level-4)

The Diabetic Retinopathy (DR) Questionnaire page in Fig. 11.6, and users are asked to respond to a series of questions about their vision, including diabetes, eye pain, hazy vision, floaters, difficulties seeing at night, and fluctuating vision. They can click the "Proceed to Prediction" button to move on to the prediction step based on their answers.

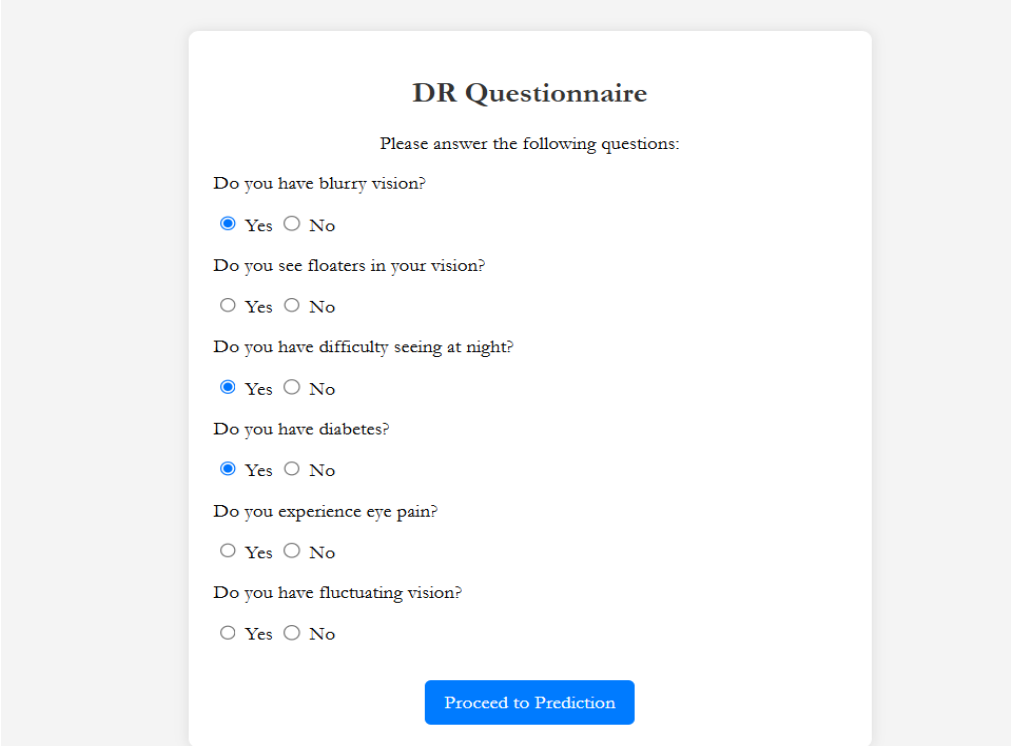

Fig 11.6 Questionnaire Screen

Once classified, the output screen (Fig 11.7) provides the prediction result along with a confidence score. Additionally, the interface offers basic eye health tips based on the detected stage, ensuring users receive relevant guidance. The design ensures a smooth user experience with quick and accurate results.

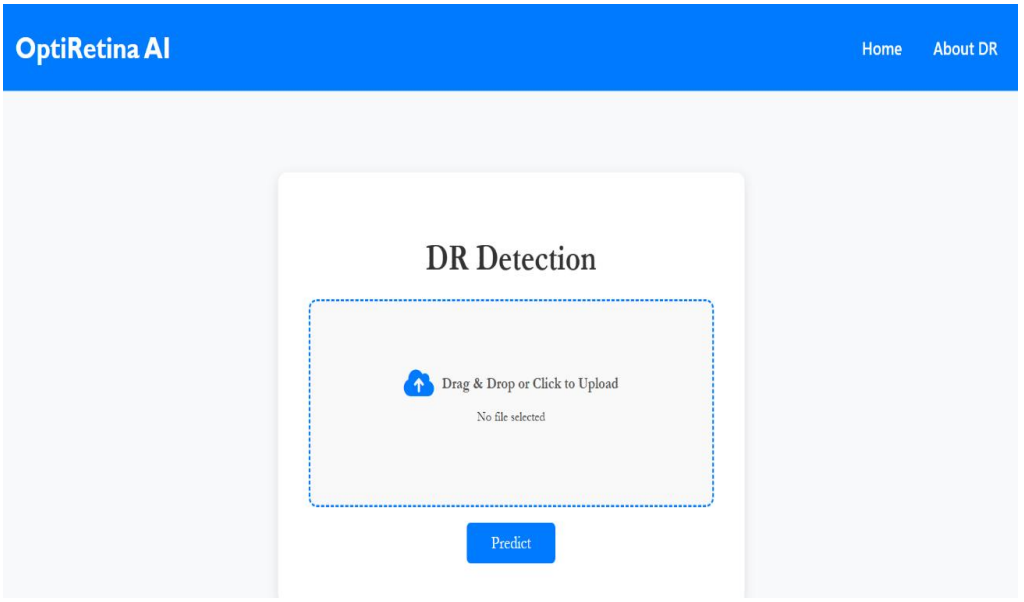

Fig 11.7 Output Screen

12. CONCLUSION

This study provides an overview of the identification of Diabetic Retinopathy (DR), taking into account the importance of early detection in preventing vision loss. DR is the main reason for blindness in persons with diabetes. This work employed an integrated Optimized Convolutional Neural Network with Gray Wolf Optimization to focus on feature selection and hyperparameter optimization. These findings unambiguously demonstrate that GWO significantly enhances CNN performance by lowering complexity, raising accuracy, and improving other metrics like feature selection to only pertinent ones from a high dimensional dataset. Simultaneously, the GWO-CNN model outperformed the other models for the DR dataset, with an accuracy of 99.31%. This demonstrates that GWO is among the best models in contemporary computational approaches for Deep Learning model optimization, making it perfect for intricate medical diagnoses such as deep reinforcement learning. The technique developed in this work not only reduces the workload for medical staff but also ensures timely and accurate diagnosis, thereby improving the outcome of the patient. Since this technique is more sensitive, highly accurate, and can be integrated easily into clinical practice, it holds great promise for enhancing medical diagnostics in the era of computational medicine.

13. FUTURE SCOPE

In future studies, Ant Colony Optimization (ACO) could be used for feature selection and for hyperparameter tuning because it can handle larger feature sets and optimize the Convolutional Neural Network (CNN) more efficiently. This may result in increased accuracy and reduced computational time. The dataset's high-resolution photos present additional issues, such as increased disk consumption and lengthier processing times due to uneven image sizes. Standardizing image dimensions or employing compression techniques could help to alleviate the issue.

However, these irregular sizes are increasing preprocessing and training times. High-resolution photographs require a lot of storage and powerful hardware, which limits their application in resource-constrained clinical settings. Furthermore, the risk of overfitting persists, especially for small or imbalanced datasets. Addressing class imbalance with synthetic data or cost-sensitive learning may improve the model's generalization.

14. REFERENCES

1. Sebti, R., Zroug, S., Kahloul, L., & Benharzallah, S. (2021, November). "A Deep Learning approach for the Diabetic Retinopathy detection". In *The Proceedings of the International Conference on Smart City Applications* (pp. 459-469). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-94191-8_37.

2. Ayala, A., Ortiz Figueroa, T., Fernandes, B., & Cruz, F. (2021). "Diabetic Retinopathy improved detection using Deep Learning". *Applied Sciences*, *11*(24), 11970**.** https://doi.org/10.3390/app112411970

3. Saxena, G., Verma, D. K., Paraye, A., Rajan, A., & Rawat, A. (2020). "Improved and robust Deep Learning agent for preliminary detection of Diabetic Retinopathy using public datasets". *Intelligence-Based Medicine*, *3*, 100022**.** https://doi.org/10.1016/j.ibmed.2020.100022

4. Jabbar, A., Liaqat, H. B., Akram, A., Sana, M. U., Azpíroz, I. D., Diez, I. D. L. T., & Ashraf, I. (2024). "A Lesion-Based Diabetic Retinopathy Detection Through Hybrid Deep Learning Model". *IEEE Access*.

5. Farooq, M. S., Arooj, A., Alroobaea, R., Baqasah, A. M., Jabarulla, M. Y., Singh, D., & Sardar, R. (2022). Untangling computer-aided diagnostic system for screening Diabetic Retinopathy based on Deep Learning techniques. *Sensors*, *22*(5), 1803. https://doi.org/10.3390/s22051803.

6. Butt, M. M., Iskandar, D. A., Abdelhamid, S. E., Latif, G., & Alghazo, R. (2022). Diabetic Retinopathy detection from fundus images of the eye using hybrid Deep Learning features. *Diagnostics*, *12*(7), 1607**.** https://doi.org/10.3390/diagnostics12071607

7. Dai, L., Sheng, B., Chen, T., Wu, Q., Liu, R., Cai, C., ... & Jia, W. (2024). A Deep Learning system for predicting time to progression of Diabetic Retinopathy. *Nature Medicine*, *30*(2), 584-594**.** https://doi.org/10.1038/s41591-023-02702-z

8. Baba, S. M., Bala, I., Dhiman, G., Sharma, A., & Viriyasitavat, W. (2024). Automated Diabetic Retinopathy severity grading using novel DR-ResNet+ Deep Learning model. *Multimedia Tools and Applications*, 1-43.

https://doi.org/10.1007/s11042-024-18434-2

9. Jagadesh, B. N., Karthik, M. G., Siri, D., Shareef, S. K., Mantena, S. V., & Vatambeti, R. (2023). Segmentation Using the IC2T Model and Classification of Diabetic Retinopathy Using the Rock Hyrax Swarm-Based Coordination Attention Mechanism. *IEEE Access***.**

10. Kallel, F., & Echtioui, A. (2024). Retinal fundus image classification for Diabetic Retinopathy using transfer learning technique. *Signal, Image and Video Processing*, *18*(2), 1143-1153**.** https://doi.org/10.1007/s11760-023-02820-8 .

11. Moturi, S., Vemuru, S., Tirumala Rao, S.N., Mallipeddi, S.A. (2023). Hybrid Binary Dragonfly Algorithm with Gray Wolf Optimization for Feature Selection. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems, vol 703. Springer, Singapore. https://doi.org/10.1007/978-981-99-3315-0 47

12. Minija, S. J., Rejula, M. A., & Ross, B. S. (2024). Automated detection of Diabetic Retinopathy using optimized Convolutional neural network. *Multimedia Tools and Applications*, *83*(7), 21065-21080. https://doi.org/10.1007/s11042-023-16204-0

13. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). "Gray wolf optimizer." “Advances in Engineering Software”, 69, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007.

14. M.Sireesha, S. N. Tirumala Rao, Srikanth Vemuru, Optimized Feature Extraction and Hybrid Classification Model for Heart Disease and Breast Cancer Prediction International Journal of Recent Technology and Engineering Vol - 7, No 6, Mar - 2019 ISSN - 2277-3878, Pages 1754 – 1772

15. Sireesha Moturi, S.N.Tirumala Rao, Srikanth Vemuru, Gray wolf assisted dragonfly-based weighted rule generation for predicting heart disease and breast cancer, Computerized Medical Imaging and Graphics, Volume 91, 2021, 101936, ISSN 0895-6111

6th International Conference on Communication and Intelligent Systems (ICCIS 2024)

Organized by

Maulana Azad National Institute of Technology (MANIT), Bhopal, India

Technically Sponsored by

Soft Computing Research Society

November 08-09, 2024

Certificate of Presentation

This is to certify that **Teja Sri Vankayala** has presented the paper titled **Enhanced Optimized CNN Based Automated Diabetic Retinopathy Detection** authored by **Sireesha Moturi, Teja Sri Vankayala, M. Mounika Naga Bhavani, Eswar Kalyani Karna, Venkayamma Akkapalli, K.V Narasimha Reddy** in the 6th International Conference on Communication and Intelligent Systems (ICCIS 2024) held during November 08-09, 2024.

Prof. Sanjay Sharma
(General Chair)

Dr. Harish Sharma
(General Chair)

SCRS\ICCIS2024\PC\610

https://scrs.in/conference/iccis2024

Enhanced Optimized CNN Based Automated Diabetic Retinopathy Detection

Sireesha Moturi[1], Teja Sri Vankayala[2], M. Mounika Naga Bhavani[3], Eswar Kalyani Karna[4], Venkayamma Akkapalli[5], and K.V. Narasimha Reddy[6]

Department of CSE, Narasaraopeta Engineering College, Narasaraopeta, Palnadu District, Andhra Pradesh, India.
vtejasri.1016@gmail.com, medurimounika4@gmail.com, eswarkalyani048@gmail.com, vyshnaviakkapalli@gmail.com, narasimhareddynec03@gmail.com
sireeshamoturi@gmail.com

Abstract. Early identification is essential to prevent serious visual impairment in diabetic patients as Diabetic Retinopathy (DR) is a main reason for blindness. In this paper, an optimal Convolutional Neural Network (CNN) model is used to propose an automated approach for categorizing the stages of DR. The pre-trained VGG16 model uses deep feature extraction for the given retinal pictures, which uses scaling and feature selection heuristics by the Grey Wolf Optimizer. Those selected features from GWO belonging to the most relevant features would also give a better boost to the classification performance along with the optimization of both hyperparameters. The proposed model will perform better than the traditional techniques based on the Precision, Recall, and F1 scores' experimental results. It has an accuracy of 99.31% on the DR dataset. The inclusion of GWO in CNN, models hold tremendous potential for use in the analysis of medical images and yields Optimized CNN, an efficient technique that is effective in improving healthcare diagnosis.

Keywords: Diabetic Retinopathy · Optimized Convolutional Neural Network (OpCoNet) · Grey Wolf Optimizer (GWO) · VGG16 · Feature Selection · Hyperparameter Tuning

1 Introduction

Diabetes-related DR can cause blindness and vision impairment. Efficient screening and diagnostic techniques are crucial due to global diabetes incidence. [1] Deep learning and AI, particularly convolutional neural net- works, can automate diagnosis and categorize retinal disorders, improving patient outcomes. This study focuses to probe and improve deep learning methods for DR identification. Recent advancements in Diabetic Retinopathy detection have significantly improved diagnostic accuracy. Techniques like deep neural networks, convolutional neural networks, deep residual networks, DenseNet, and automated methods for detecting microaneurysms have been used. These contributions inform the use of deep learning [2] algorithms for automated drug resistance identification, enhancing the accuracy of DR diagnosis. Diabetes-related Diabetic

Retinopathy (DR) is the main techniques. [3] Automated systems, using CNN, can provide a reliable tool for early diagnosis and treatment, enhancing accessibility and effectiveness in clinical settings. This study explores a hybrid deep learning strategy to refine the efficiency and accuracy of Diabetic Retinopathy categoriza- tion, a serious microvascular consequence of diabetes, and reduce the burden of this crippling condition [4], utilizing recent artificial intelligence advancements in deep learning. Researchers have developed the DeepDR Plus system, using AI to forecast the progression of Diabetic Retinopathy, a serious side effect of diabetes. This innovative approach aims to improve screening methods and personalized patient care, ultimately reducing the burden of diabetes-related visual loss. The DR ResNet plus model automates Diabetic Retinopathy severity grading and diagnosis, improving patient outcomes and enhancing diagnosis accuracy using large datasets. This study explores novel models and strategies for efficient segmentation and classification of Diabetic Retinopathy (DR), the main basis of vision loss in diabetes patients, using machine learning and also it investigates the effectiveness of pre-trained models like VGG16, VGG19 in improving diagnostic precision for DR a major cause of vision impairment.

2 Related Work

Some deep learning models include VGG, ResNet, and DenseNet. These deep learning models were found to suffer from gaps in diabetic retinopathy detection. There is a need to optimize the Convolutional Neural Network using methods such as Grey Wolf Optimization for increasing the precision of detection. Shankar et al. have used deep learning algorithms and achieved 99.28% accuracy for diabetic retinopathy diagnosis [1]. Other models did less well, such as ResNet34 at 86% sensitivity and 85% accuracy and decision trees at 91% and VGGNet at 92%. Gadekallu et al. used CNNs to predict diabetic retinopathy that reached an accuracy of 96% with feature extraction [2]. Gaurav Saxena hyperparameter-optimized InceptionResNetV2 that achieved an AUC of 0.92 on the Messidor-2 dataset [3].

Hybrid model which combines ResNet and GoogleNet with adaptive particle swarm optimization achieved 94 percent accuracy by Ayesha Jabbar, et al. authors, which approaches demographic bias and reports on accuracy, precision, recall, and F1 score [4]. Farooq's CNN achieves 73%, Shah's hybrid CNN-RNN gets 90%, and Ali's transfer learning model obtains 85%[5]. Javed's optimization produced DR diagnosis automated up to an accuracy of 88%, while Butt's modified VGGNet showed 83.1% accuracy for the EyePACS dataset [6]. Iskandar's DenseNet, consisting of an attention module, obtained 97%, whereas Latif's meta-plasticity approach achieved an accuracy of 94%. Bin Sheng's DeepDR Plus system, in predicting DR, used a range of C-index between 0.823 and 0.862, so it demonstrates its applicability in the early detection of DR [7].

Gaurav Dhiman's optimized DR-ResNet+ achieved an accuracy of 98.98% with 98.29% sensitivity and 99.16% specificity using the grid and random search techniques [8]. The IC2T model developed by Jagadesh et al. was able to attain

98% accuracy in diabetic retinopathy detection depending on the features of the biomarkers that include blood vessels and optic discs along with a specific effect of the model architecture and optimization of the hyperparameters [9].

3 Proposed Method

The Methodology provides a detailed structure for the classification of Diabetic Retinopathy. This methodology contains a sequence of steps that help to build the classifier easily. The Fig.1, describes the steps for the DR Detection

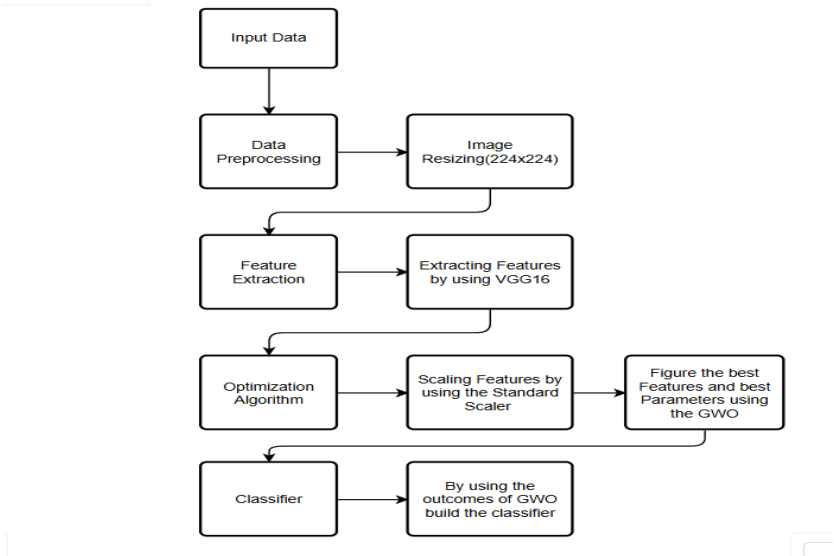

Fig. 1. The Structure of DR Classification

3.1 Dataset Description

The "Diabetic Retinopathy Level Detection" dataset on Kaggle includes retinal images categorized into different levels of severity of diabetic retinopathy. The present research used 4396 images to classify the level of DR. The Complete process divides the photographs into the Training and Validation sets. The Fig. 2, explains that the DR Photographs are categorized into 5 Levels.

DR of (Level 0) is absent No symptoms of diabetic retinopathy are present. The retina seems to be in normal condition. (Level 1) Mild NPDR is characterized by microaneurysms, which are tiny blood vessel bulges in the retina. There are no further retinopathy symptoms. Diabetic Retinopathy with Moderate Non-Proliferative Effects (Level 2) There are more microaneurysms along with other

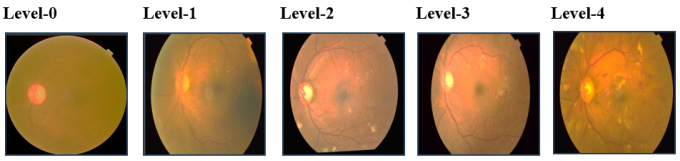

Fig. 2. Severity Of DR

abnormalities such as retinal hemorrhages and exudates. There can be some anomalies in the blood vessels as well. Severe DR with No Proliferation (Level 3) More substantial alterations are present at this level, including several hemorrhages in the retina Venous beading (veins twisted and dilated) anomalies of the intraretinal microvascular system (IRMA) The likelihood of developing proliferative Diabetic Retinopathy rises at this point. Diabetic Proliferative Retinopathy (Level 4) is distinguished by the development of new blood vessels on the optic disc or retina (neovascularization). This stage can lead to serious complications, including vitreous hemorrhages and retinal detachment, which can cause permanent vision loss if not treated. A total of 3662 Images are used for training the

Fig. 3. Class Imbalance of images in Training Image Dataset

optimized convolutional neural network [1][10]. Most images belong to no DR Stage representing the Class Imbalance. Fig. 3, represents the Class Imbalance of the photographs

3.2 Experimental Setup

The experimental setup used in this work is the Google Colaboratory. This Google Colab provides cloud-based services with TPU and GPU computational resources. The TPU v2-8 has larger RAM at 334.66 GB and 100 GB of disk storage, hence it executes faster. Hardware Description: AMD Ryzen 5000 with

16 GB RAM was used in conducting this study to build the model in the images. NVIDIA denotes the GPU that is currently installed on the system.

3.3 Data Preprocessing

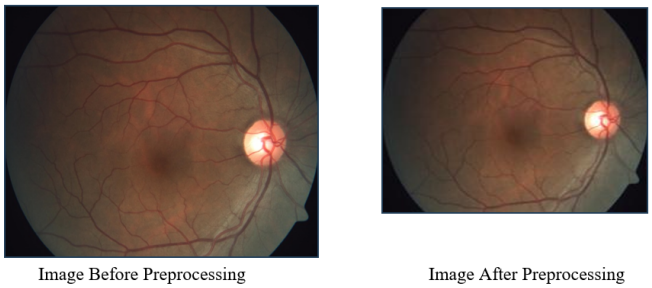

Fig. 4. The Comparison of Image Before and After Preprocessing

The Diabetic Retinopathy Level Detection dataset is made up of high quality images. To speed up the training process, the images were resized to 224x224 and processed in batches of 32 to ensure consistency and conformity with the VGG16 model used for feature extraction. Fig. 4, outlines the differences before and after preprocessing. These methods improve computational efficiency and the model's ability to identify features when classifying diabetic retinopathy. Since the dataset was already divided to accommodate both the training and testing sets, data splitting was not required. Class weights are computed to address an issue referred to as class imbalance [8] this prevents the model from being biased toward the majority class and thus removes the overuse of one's respective sampling of data to overcome this phenomenon. No data augmentation techniques, such as rotations and flips, were performed in the process, and instead solely relied upon the given preprocessing procedures.

3.4 Feature Extraction

This process uses the pre-trained VGG16 convolutional neural network (CNN) [14], which is renowned for its depth and simplicity, to extract deep information from photos. Because it can extract high-level abstract features from images, the Oxford Visual Graphics Group developed VGG16, a 16-layer system with learnable weights that is frequently used for image classification. Our special dataset, which consists of image training and validation directories, was subjected to the VGG16 model.

The Images were prepared by scaling them between 0 and 1 and normalizing their pixel values by dividing them by 255.0. This facilitates training at a faster convergence rate. To guarantee correct scaling and normalization, the VGG16-specific preprocessing function preprocess input was also used. As requested by

VGG16, we resized the batches of photos to 224x224 pixels using the Image Data Generator, along with the corresponding labels.

The convolutional basis of the VGG16 model was utilized to extract deep features from the images using pre-trained weights from the ImageNet dataset. Then, features were combined into one array by using numpy.save() with their labels. This speeds up classifier training because it is not necessary to repeatedly extract features. Because the same stored features could be used for other applications, such as model optimization and training machine learning classifiers, thousands of computation time will be saved in total.

3.5 Grey Wolf Optimization

The Grey Wolf Optimizer (GWO) is an optimization algorithm inspired by nature, designed to replicate the hierarchical social structure and cooperative hunting strategies of grey wolves in their natural habitat. Mirjalili et al. introduced it in 2014. The primary purpose of GWO is to expedite the search space and compare the grey wolf hunting process to determine the best solution to a problem. GWO does an excellent job of optimizing parameters [11][12] Feature selection extracts the most significant features from high-dimensional datasets, simplifying and improving model performance. Global Optimization Finding the global optimum for a particular problem while avoiding local optimal traps is critical in complicated or nonlinear systems [13][15].

We extracted features from our dataset, resulting in a high dimensional feature space with 25,088 features per Image. Feature Scale We applied a standard scaling method to the extracted data from our training and validation sets to ensure a consistent comparison of features. To narrow down the criteria and select only the most important features, we used GWO. A feature selection mask was then applied to the scaled input data. This mask acts as a binary filter: a value of '1' indicates that the feature is selected, while '0' means it is excluded. The decision to include or exclude each feature is guided by the GWO algorithm, which evaluates the contribution of each feature to the model's performance. Class homogeneity We calculated the class weights to check for the homogeneity of the data set so that the CNN model is not biased toward one class. The CNN model is designed with two connected layers (thick). In addition, a feature selection mask was used to select the most relevant features from the input data. The objective function assessed the model's performance in the validation set by calculating the validation accuracy. The most important optimal parameters are:

- Number of neurons in the first dense layer
- Number of neurons in the second layer
- Learning rate
- Dropout rate
- Selected features

This algorithm considers three variables Alpha (α), Beta (β), and Delta (δ) come from. The rest of the scenarios for these wolves show possible solutions, and the hunting process is compared as an optimization process.

- Alpha wolves (α) lead the hunt, representing the best solutions found so far.
- Beta (β) and Delta (δ) wolves help Alpha and explore the search area properly.
- Omega wolves (ω), representing the rest of the population, follow these leads and converge to the optimal solution.

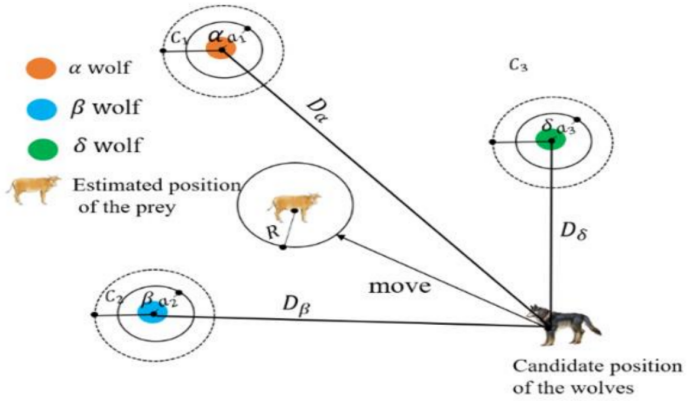

Fig. 5. Schematic Representation of the Grey Wolf Optimizer's Hunting Behavior

The new position of a wolf is calculated as the average of three positions determined by the influence of the alpha, beta, and delta wolves. Fig. 5, provides how calculations will be done in the positions

$$Z(t+1) = \frac{Z_1 + Z_2 + Z_3}{3} \tag{1}$$

Each of the positions Z_1, Z_2, and Z_3 is calculated using the following equations

$$Z_1 = \alpha(t) - P_1 \cdot A_\alpha \tag{2}$$

$$Z_2 = \beta(t) - P_2 \cdot A_\beta \tag{3}$$

$$Z_3 = \delta(t) - P_3 \cdot A_\delta \tag{4}$$

The distance between a wolf's position and the prey (best solution) is calculated as

$$A_\alpha = |Q_1 \cdot \alpha(t) - Z(t)| \tag{5}$$

$$A_\beta = |Q_2 \cdot \beta(t) - Z(t)| \tag{6}$$

$$A_\delta = |Q_3 \cdot \delta(t) - Z(t)| \tag{7}$$

The coefficients P and Q are calculated using the following formulas

$$P = 2 \cdot b \cdot s_1 - b \tag{8}$$

$$Q = 2 \cdot s_2 \tag{9}$$

Where s1 and s2 are unplanned numbers in the range [0, 1]. The parameter α decreases linearly throughout iterations to balance exploration and exploitation

$$a = 2 - \frac{2 \cdot t}{T} \tag{10}$$

t is the current iteration.
T is the total number of iterations allowed.

These equations are fundamental to the operation of the GWO algorithm, guiding the wolves toward the optimal solution in the search space. We performed GWO for ten wolves over 20 iterations of movement, where they moved based on their distance to the optimum solution. The algorithm selected almost 10,316 features out of the original 25,088, reducing the dimensionality and performance of the model. The GWO method effectively learned suitable hyperparameters along with feature selection that positively improved the CNN model at a relatively lower computing cost. This approach may be applied with even more large-scale high-dimensional datasets for machine learning problems.

3.6 Optimized Convolutional Neural Network

OpCoNet is a designated deep learning model for the classification of stages of diabetic retinopathy. Herein, the model made use of CNNs with some advanced optimization techniques to enhance its performance. For hyperparameter tuning, OpCoNet adopted GWO which is inspired by the natural leadership and hunting strategy of grey wolves. GWO helps optimize the parameters of the network while choosing the features that are most relevant for the task at hand from high-dimensional data of images improving accuracy together with efficiency.

The model starts with an input layer that computes selected features from a dataset. Then comes the dense layer, where neurons perform weighted sums and apply activation functions to learn complex patterns. To avoid overfitting, methods such as L2 regularization should be applied to ensure that the model generalizes well beyond the training data.

Table 1. The hyperparameters for the model construction.

Hyperparameters	**Values**
Neurons Layer 1	181
Neurons Layer 2	150
Learning Rate	0.000457
Optimizer	Adam
Dropout Rate	0.1
L2 norm regularization	0.001
Loss	Categorical Cross Entropy

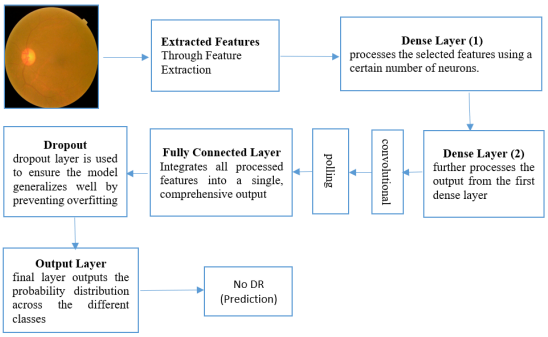

Fig. 6. The Model Architecture

Then, OpCoNet utilizes a batch normalization layer stabilizing and speeding up the training: it normalizes the output of the dense layer. Dropout: neurons are randomly disabled in training for preventing overfitting.

This is repeated in a second dense layer with a different number of neurons and batch normalization followed by dropout as shown in Fig. 6. Then, a convolutional layer applies filters to actually capture complex patterns existing within the data. Shrinkage-based pooling preserves important information within this step, thus improving efficiency. The final layer is a fully connected output layer with a SoftMax activation function producing a probability distribution to classify input.

The key hyperparameters listed in Table 1, including the number of neurons in each layer, dropout rate, learning rate, and L2 norm regularization, were determined based on the outputs of the Grey Wolf Optimizer (GWO) combined with experimental analysis. To train the model, it employs an optimizer where it calculates the weights from categorical cross-entropy loss. Early stopping has also been implemented to prevent overtraining. This stops the model at that point with no further development and applies a well-balanced and accurate predictive model.

4 Result

The best hyperparameters of the Grey Wolf Optimization build the Optimized CNN which accepts the Extracted Features as input and also the Images. The Optimized CNN with GWO Selected Features gives greater results concerning evaluation metrics. The OpCoNet with GWO Selected Features takes the Selected Extracted Features as input. Each image may have around 25088 features using the GWO, the feature set was reduced to 10,316 features. The Optimized CNN With GWO Selected Features gains 99.31% accuracy over the 3662 images Extracted Features Fig. 7, shows the classification report of the OpCoNet on the validation extracted features. Fig. 8, shows the classifier works better on the Validation Dataset

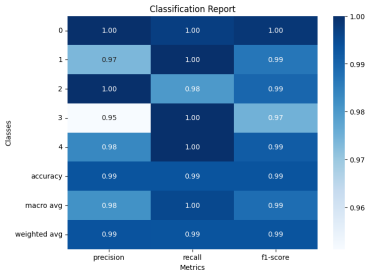

Fig. 7. Classification Report on the Validation Dataset

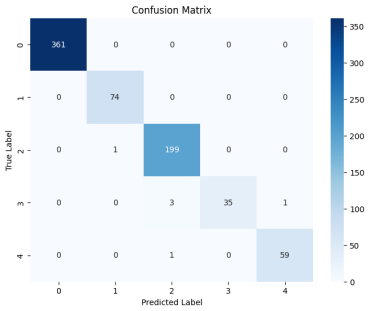

Fig. 8. Confusion Matrix on the Validation Dataset

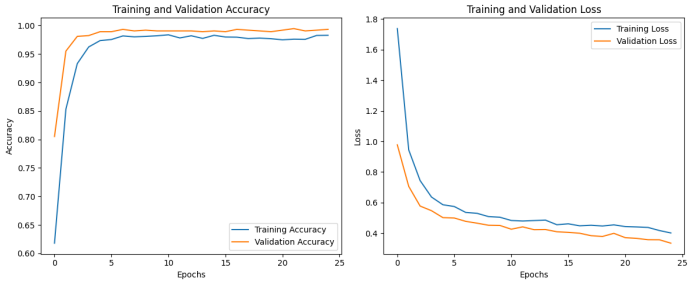

Fig. 9. The Model Performance Plots

The Training and Validation accuracy, loss from Fig. 9. The graphs illustrate strong convergence, with both training and validation accuracy nearing 99.50% while training and validation loss significantly decrease, indicating the model is well-trained without evident overfitting. Table. 2, compares different methodolo-

Table 2. Comparison of Methodologies with Different Feature Selection Approaches

Methodology	**No of Features Selected**	**Epochs**	**Accuracy**	**Time Comparison**
OpCoNet With Images (GWO) [12]	Nill	25	98.23	3 Hours
OpCoNet With All Extracted Features (GWO)	All (25088)	30	99.18	48.221 Sec
OpCoNet With GWO Selected Features	**10316**	**25**	**99.31**	**27.279 Sec**
OpCoNet With ACO Selected Features	12565	25	99.45	28.366 Sec
CNN With Images [1]	Nill	25	96.03	3 Hours
OpcoNet With All Extracted Features (ACO)	All (25088)	25	98.91	76.799 Sec

gies based on selected features, epochs, accuracy, and model-building time. The OpCoNet model with GWO-selected features achieves a remarkable accuracy

of 99.31% in 25 epochs, with a fast training time of just 27.279 seconds. This highlights GWO's effectiveness in enhancing both accuracy and computational efficiency. In comparison, the OpCoNet model with Ant Colony Optimization-Selected Features achieves a slightly higher accuracy of 99.45% but takes longer (28.366 seconds). Models using all extracted features (99.18%) or image-based methods (98.23%) deliver lower accuracy and are more computationally demanding, with the image-based approach taking up to 3 hours for training. The OpcoNet with all extracted features using the best parameters of ACO achieves a lower accuracy(98.91%) and takes a long time to build the model. Among the various methods tested, the GWO-selected features demonstrate the best trade-off between accuracy and computational efficiency, making it the most effective approach.

5 Conclusion

This study provides an overview of the identification of diabetic retinopathy (DR), taking into account the importance of early detection in preventing vi sion loss. DR is the main reason for blindness in persons with diabetes. This work employed an integrated Optimized Convolutional Neural Network with Grey Wolf Optimization to focus on feature selection and hyperparameter op timization. These findings unambiguously demonstrate that GWO significantly enhances CNN performance by lowering complexity, raising accuracy, and im proving other metrics like feature selection to only pertinent ones from a high dimensional dataset. Simultaneously, the GWO-CNN model outperformed the other models for the DR dataset, with an accuracy of 99.31%. This demonstrates that GWOis among the best models in contemporary computational approaches for deep learning model optimization, making it perfect for intricate medical diagnoses such as deep reinforcement learning. The technique developed in this work does not only reduces the workload for medical staff but also ensures timely and accurate diagnosis, thereby improving the outcome of the patient. Since this technique is more sensitive, highly accurate, and can be integrated easily into clinical practice, it holds great promise for enhancing medical diagnostics in the era of computational medicine

6 Dataset Availability

The Dataset link is `https://www.kaggle.com/datasets/arbethi/diabetic-retinopathy-level-detection`

References

1. Sebti, R., Zroug, S., Kahloul, L., & Benharzallah, S. (2021, November). "A deep learning approach for the diabetic retinopathy detection". In *The Proceedings of the International Conference on Smart City Applications* (pp. 459-469). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-94191-8_37

2. Ayala, J. A., Ortiz Figueroa, T., Fernandes, B., & Cruz, F. (2021). "Diabetic retinopathy improved detection using deep learning". *Applied Sciences*, 11(24), 11970. https://doi.org/10.3390/app112411970
3. Saxena, G., Verma, D. K., Paraye, A., Rajan, A., & Rawat, A. (2020). "Improved and robust deep learning agent for preliminary detection of diabetic retinopathy using public datasets". *Intelligence-Based Medicine*, 3, 100022.
4. Jabbar, A., Liaqat, H. B., Akram, A., Sana, M. U., Azpíroz, I. D., Diez, I. D. L. T., & Ashraf, I. (2024). "A Lesion-Based Diabetic Retinopathy Detection Through Hybrid Deep Learning Model". *IEEE Access*.
5. Farooq, M. S., Arooj, A., Alroobaea, R., Baqasah, A. M., Jabarulla, M. Y., Singh, D., & Sardar, R. (2022). "Untangling computer-aided diagnostic system for screening diabetic retinopathy based on deep learning techniques". *Sensors*, 22(5), 1803.
6. Butt, M. M., Iskandar, D. A., Abdelhamid, S. E., Latif, G., & Alghazo, R. (2022). "Diabetic retinopathy detection from fundus images of the eye using hybrid deep learning features". *Diagnostics*, 12(7), 1607. https://doi.org/10.3390/diagnostics12071607
7. Dai, L., Sheng, B., Chen, T., Wu, Q., Liu, R., Cai, C., & Jia, W. (2024). "A deep learning system for predicting time to progression of diabetic retinopathy". *Nature Medicine*, 30(2), 584-594. https://doi.org/10.1038/s41591-023-02702-z
8. Baba, S. M., Bala, I., Dhiman, G., Sharma, A., & Viriyasitavat, W. (2024). "Automated diabetic retinopathy severity grading using novel DR-ResNet+ deep learning model". *Multimedia Tools and Applications*, 1-43.
9. Jagadesh, B. N., Karthik, M. G., Siri, D., Shareef, S. K., Mantena, S. V., & Vatambeti, R. (2023). "Segmentation Using the IC2T Model and Classification of Diabetic Retinopathy Using the Rock Hyrax Swarm-Based Coordination Attention Mechanism". *IEEE Access*.
10. Kallel, F., & Echtioui, A. (2024). "Retinal fundus image classification for diabetic retinopathy using transfer learning technique". *Signal, Image and Video Processing*, 18(2), 1143-1153.
11. Moturi, S., Vemuru, S., Tirumala Rao, S.N., Mallipeddi, S.A. (2023). Hybrid Binary Dragonfly Algorithm with Grey Wolf Optimization for Feature Selection. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems, vol 703. Springer, Singapore.https://doi.org/10.1007/978-981-99-3315-0_47
12. Minija, S. J., Rejula, M. A., & Ross, B. S. (2024). "Automated detection of diabetic retinopathy using optimized convolutional neural network". *Multimedia Tools and Applications*, 83(7), 21065-21080.https://doi.org/10.1007/s11042-023-16204-0
13. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). "Grey wolf optimizer". *Advances in Engineering Software*, 69, 46-61.
14. M.Sireesha, S. N. Tirumala Rao, Srikanth Vemuru, Optimized Feature Extraction and Hybrid Classification Model for Heart Disease and Breast Cancer Prediction International Journal of Recent Technology and Engineering Vol - 7, No 6, Mar - 2019 ISSN - 2277-3878, Pages – 1754 – 1772
15. Sireesha Moturi, S.N.Tirumala Rao, Srikanth Vemuru, Grey wolf assisted dragonfly-based weighted rule generation for predicting heart disease and breast cancer, Computerized Medical Imaging and Graphics, Volume 91, 2021, 101936, ISSN 0895-6111.

ORIGINALITY REPORT

8%	4%	6%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Innovative Computing and Communications", Springer Science and Business Media LLC, 2024 Publication	1%
2	S. Jasmine Minija, M. Anline Rejula, B. Shamina Ross. "Automated detection of diabetic retinopathy using optimized convolutional neural network", Multimedia Tools and Applications, 2023 Publication	1%
3	www.ncbi.nlm.nih.gov Internet Source	<1%
4	"Data Science and Applications", Springer Science and Business Media LLC, 2024 Publication	<1%
5	dokumen.pub Internet Source	<1%
6	Submitted to University of Southampton Student Paper	<1%