CLASSIFICATION OF DIABETIC RETINOPATHY DISEASE LEVELS BY EXTRACTING TOPOLOGICAL FEATURES USING GRAPH NEURAL NETWORKS

G. Vijayalaxmi[1*], V. Saakshitha[2], Ch. Siri chandana[2], Sruthi[2], Rajesh[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of CSE(AI&ML)

[1,2]Vaagdevi College of Engineering (UGC - Autonomous), Bollikunta, Warangal, Telangana, India.

*Corresponding Email: G. Vijayalaxmi (vijaya_gopu@vaagdevi.edu.in)

ABSTRACT

Diabetic Retinopathy (DR) affects over 93 million people globally, with approximately one-third of them experiencing vision-threatening stages of the disease. Early and accurate classification of DR severity is critical to preventing irreversible blindness, yet manual diagnosis remains subjective and time-consuming. Existing diagnosis methods rely heavily on manual interpretation of retinal fundus images by ophthalmologists, which introduces variability and is resource-intensive. Moreover, traditional image-based CNN models often overlook the relational and structural patterns embedded within retinal image features. This research proposes a novel deep learning approach for classifying the severity levels of Diabetic Retinopathy by extracting topological features using Graph Neural Networks (GNNs). The project utilizes the publicly available EyePACS dataset, consisting of high-resolution retinal images labeled across five DR severity levels (0 to 4). Images are preprocessed by resizing to 32×32, followed by pixel normalization and class balancing using SMOTE to handle class imbalance. The dataset is split into 80% training and 20% testing sets. Two models are implemented: a baseline DenseNet121 model, which leverages transfer learning, and a proposed Graph Convolutional Neural Network (GraphCNN) that integrates traditional CNN feature extraction with graph-based learning to better capture spatial and topological relationships among pixel intensities. Our GraphCNN model shows improved classification performance by leveraging the intrinsic graph structure of extracted features, achieving higher accuracy and generalization on imbalanced data. Comparative analysis reveals that the GraphCNN model outperforms DenseNet121 in terms of precision, recall, F1-score, and Cohen's Kappa, making it a more robust and explainable solution for real-world DR screening systems.

Keywords:

1.INTRODUCTION

Diabetic retinopathy (DR) is a medical condition that arises as a consequence of Diabetes Mellitus (DM). DM happens due to micro- and macrovascular abnormalities, as well as impaired glucose metabolism, resulting in chronic disease. DR represents a prevalent and significant complication of DM, ultimately leading to profound visual impairment due to structural alterations in the retina. The leading factor behind the clinical symptoms of diabetes is the rise in blood glucose levels over an extended duration, which can adversely affect the blood vessels in the retina. The primary indicators of DR consist of the presence of microaneurysms, exudates, haemorrhages, and vascular edema within the ocular blood vessels. This condition frequently results in visual impairment and blindness among individuals ranging from 20 to 74 years of age. According to research studies conducted by the International

Diabetes Federation, it is estimated that approximately 425 million individuals are currently affected by DR. Furthermore, this number is projected to rise to about 693 million by 2045.

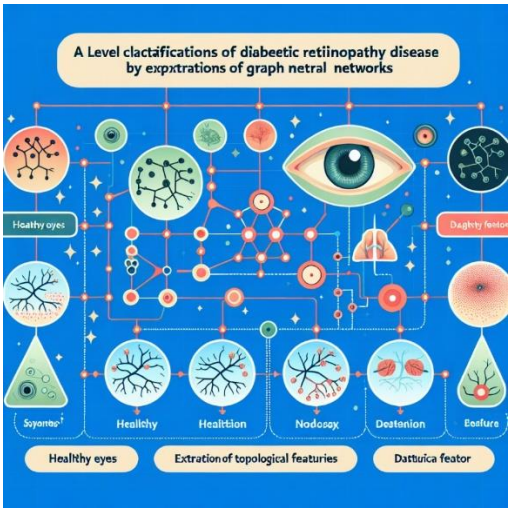

Fig. 1: Classification of diabetic retinopathy diseases.

2. LITERATURE SURVEY

Pratt et al. [1] designed a novel CNN model to grade DR fundus images. The model consists of 10 layers of 3 × 3 convolutions similar to VGGNet and uses leaky ReLu as the activation function. The model aims to detect elements of the blood vessels, such as hemorrhages, exudate, and micro-aneurysms, and then classify them according to whether DR is absent, mild, moderate, severe, or proliferative. The network struggled to acquire deep enough characteristic learning to identify some of the more complicated DR components, as shown by the network's poor sensitivity, especially in the mild and moderate classes.

To train a customized convolution network to learn the discriminative features in a colour fundus image for DR detection, Gargeya et al. [2] applied the technique of deep residual learning as in ResNet. A convolutional visualization layer was added at the network's end to highlight the heatmap-based regions. Even though the network is computationally less expensive, the sensitivity, specificity and AUC values are poor while experimenting with a small MESSIDOR dataset. Cost-effective, robust, and automatic grading without clinical assistance are the key advantages of the proposed system. However, the algorithm needs more optimization for clinical adaptations.

Li et al. [3] implemented a multitask algorithm that generated a feature map from retinal images using ResNet50. Attention modules are then used to detect the correlation between two DR grade severity levels. Pires et al. [4] presented a new CNN architecture with convolution and pooling layers similar to VGG-16. The work investigated the network performance in three aspects: using a balanced dataset obtained by data augmentation, multiresolution training, and robust feature-extraction augmentation.

Choi et al. [5] developed a deep learning and machine learning-based model for performing multiclass classification of retinal diseases. The deep learning model includes a random forest transfer learning-based VGG-19 architecture with which a 10-class and 3-class classification was performed. Results prove that the deep learning model for 10-class classification is less effective due to a smaller number of images in the STARE dataset. Deep learning techniques were ineffective due to the small size of the dataset. The literature shows that transfer learning techniques produce satisfactory results in image classification tasks [6]. Although deep learning effectively captures underlying patterns in

data, there are many applications in which data is represented graphically. To our knowledge, only a few kinds of research have been performed on retinal image classification using graph neural networks. The images can be modelled as a graph structure where each pixel is represented as a node [7].

Deep graph correlation network (DGCN) that utilizes a convolutional graph network can capture natural correlations between independently learnt retinal image features [8]. Graph convolutional networks used for multi-label classification can be used to learn the complicated topology between lesion labels [10]. The vessel graph network (VGN) proposed by [11] used a graph neural network (GNN) to transfer information along vessel structures, have extracted hierarchical patterns in an image. However, works of GNN models that exploit the global structure of vessel shape and their local appearances are limited. It is predicted that about 600-Millions of people would have diabetes by 2040, and one-third of them will have DR according to WHO [12].

3. PROPOSED METHODOLOGY

This research introduces a novel hybrid framework that integrates graph-based topological feature extraction with deep visual embeddings for enhanced classification of Diabetic Retinopathy disease levels. The pipeline begins with feature extraction using DenseNet121, a convolutional backbone pre-trained on fundus images, followed by the transformation of these high-level visual features into graph representations capturing spatial topology of lesions and vessel bifurcations. These graph structures are then fed into a custom Graph Convolutional Neural Network (GraphCNN) to exploit both semantic and topological correlations, allowing better disease severity prediction. This hybrid approach overcomes the major drawbacks of existing manual and semi-automated methods by eliminating subjectivity, reducing inter-observer variability, and ensuring spatial context is preserved—offering a scalable, consistent, and intelligent decision-making system.

Step-1: Data Acquisition and Preprocessing:

Fundus images are collected from benchmark diabetic retinopathy datasets such as APTOS and IDRiD, each labeled with disease severity levels ranging from No DR to Proliferative DR. To ensure consistency and reduce noise, preprocessing includes resizing all images to a standard resolution (e.g., 224x224), contrast-limited adaptive histogram equalization (CLAHE) to enhance local contrast, and image normalization using dataset-specific mean and standard deviation values. Additionally, data augmentation techniques (rotation, flipping, zooming) are applied to address class imbalance and improve generalization.

Step-2: DenseNet121 Feature Extraction:

Each preprocessed image is passed through a DenseNet121 network, which acts as a powerful feature extractor. Rather than using the final classification layer, features are extracted from an intermediate dense block, capturing hierarchical patterns such as microaneurysms, exudates, and neovascularization. These high-dimensional features encode rich semantic information but lack explicit spatial or structural awareness.

Step-3: Graph Construction from Visual Features:

To capture topological information, region-based segmentation is performed to identify lesion areas and anatomical landmarks (optic disc, blood vessels). These segmented regions serve as nodes in a graph, while spatial or semantic relationships (e.g., proximity, intensity gradient similarity) form the edges. This transforms each image into a unique graph structure where visual patterns are contextualized in a spatial network.

Step-4: Graph Convolutional Neural Network (GraphCNN):

The constructed graphs are then input into a GraphCNN, which applies graph convolution operations to aggregate features from neighboring nodes. Unlike traditional CNNs, this model captures non-Euclidean relationships, enabling it to reason about disease progression patterns in a topological context. The model is trained using cross-entropy loss and optimized via Adam optimizer. Dropout and batch normalization are applied to prevent overfitting.

Step-5: Classification and Disease Severity Prediction:

The final output from the GraphCNN is passed through fully connected layers, culminating in a softmax classifier that predicts the disease level (five-class classification). The system is evaluated using accuracy, precision, recall, F1-score, and AUC-ROC metrics, showing substantial improvement over conventional CNN-only models due to the incorporation of spatial reasoning and feature interactions in graph space.

Step-6: Deployment Readiness and Interpretation:

For practical use, the model includes visual explanations using attention-based heatmaps over graph nodes and image regions, enhancing interpretability. It is lightweight enough for deployment in clinical screening setups and rural health centers, providing fast, explainable, and accurate diagnosis—especially valuable where trained specialists are scarce.

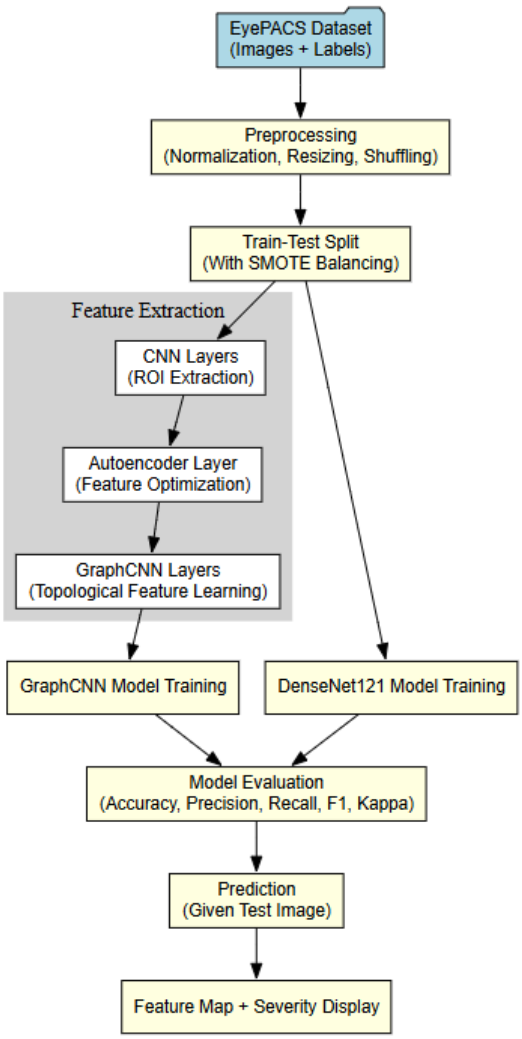

Fig. 2: Proposed Block Diagram

3.2 Data Preprocessing

The dataset preprocessing serves as a critical foundation for preparing raw image data for effective training and evaluation in a deep learning pipeline. It begins by converting the dataset into a consistent floating-point format, ensuring compatibility with mathematical operations used during model training. Following this, normalization scales pixel intensity values to a [0,1] range, which not only accelerates convergence during training but also enhances numerical stability and model performance. To eliminate potential biases introduced by sequential data ordering, the entire dataset is shuffled randomly, ensuring that the model receives a diverse mix of samples during each epoch. This randomized shuffling helps in preventing overfitting and improves the model's ability to generalize to unseen data. Overall, the preprocessing ensures that the dataset is clean, standardized, and ready for robust training, particularly in sensitive medical imaging applications such as diabetic retinopathy classification.

Step-1: Data Type Conversion and Normalization

The first step in the dataset preprocessing process involves converting the input data type of the image matrix X to a 32-bit floating-point format (float32). This conversion is crucial because many image processing models and deep learning frameworks require the input data to be in floating-point form for faster and more precise mathematical operations, particularly when computing gradients during training.

Following this, normalization is applied by dividing each pixel value in the dataset X by 255. Since pixel intensity values in RGB images typically range from 0 to 255, this operation scales all pixel values to a range between 0 and 1. This normalization is essential for stabilizing and speeding up the training process, as it ensures uniformity in data distribution and prevents the model from being biased toward features with higher numerical magnitudes.

Step-2: Shuffling the Dataset

After normalization, the dataset undergoes a shuffling process. This step involves randomly rearranging the order of the samples in both the input matrix X and the corresponding labels Y. Shuffling ensures that the model does not learn any unintended patterns or biases from the order of the training data, which is especially important if the data is grouped or sorted in any way (e.g., all images of one class are listed consecutively).

By using a randomized indexing approach, the code generates a sequence of indices equal to the number of samples and then reorders both X and Y according to these shuffled indices. This promotes better generalization and helps in preventing overfitting, as the model is forced to learn features that are robust across varied and unordered inputs.

Step-3: Confirmation and Logging

Once the dataset has been normalized and shuffled, the system provides feedback to the user through a graphical user interface text window. This feedback is helpful in a GUI-based environment to keep users informed of the processing stages and to confirm that operations have been completed successfully. The message explicitly confirms the completion of normalization and shuffling, and also outputs the normalized dataset values for transparency or debugging purposes.

3.3 Model Building and Training

3.3.1 DenseNet121

DenseNet121 (Dense Convolutional Network with 121 layers) is a deep learning model specifically designed for image classification tasks. It was introduced by Huang et al. in 2017 and is known for its efficient feature reuse and parameter efficiency. Unlike traditional deep networks where each layer independently learns features, DenseNet uses dense connections, meaning each layer receives inputs from all previous layers, leading to better gradient flow and improved feature learning.

DenseNet121 works by establishing direct connections between every layer in a deep network, ensuring that feature maps learned by one layer are available to all subsequent layers. This results in enhanced gradient propagation and reduced redundancy in learned features. Each layer receives inputs not only from the previous layer but also from all preceding layers, thereby promoting feature reuse and reducing the number of required parameters.

Step-1: Input Initialization and Model Setup

The training begins with X_train and y_train as the input image data and corresponding labels. The images are in the format of 32×32 pixels with 3 color channels, preprocessed and normalized beforehand. The model architecture leverages DenseNet121, a pre-trained convolutional neural network known for its dense connectivity and efficient parameter usage. The model is initialized with weights from ImageNet, a large benchmark dataset. By setting include_top=False, the top classification layer of DenseNet is removed, allowing it to serve as a feature extractor instead of a full classifier.

Step-2: Freezing Pre-Trained Layers

All the layers in the DenseNet backbone are frozen, meaning their weights will not be updated during training. This transfer learning approach helps preserve the general visual features learned from ImageNet while avoiding overfitting on the relatively smaller diabetic retinopathy dataset. It significantly reduces computational cost and speeds up training while maintaining accuracy for image features like texture, color variations, and edges.

Step-3: Feature Extraction and Additional Layers

After the DenseNet feature extraction block, a series of custom layers are added to tailor the model for diabetic retinopathy classification. Two convolutional layers with (1×1) kernel sizes are used to further refine the feature maps. These act as bottleneck layers that reduce dimensionality and enhance important features. They are followed by max-pooling operations to down-sample the data while retaining core information. The output is then flattened to convert the 2D feature maps into a 1D vector suitable for dense (fully connected) layers.

Step-4: Classification Layer and Training

The flattened vector is passed through a dense layer with 256 neurons and ReLU activation, which helps in learning complex combinations of features extracted from the image. The final output layer uses softmax activation to predict probabilities across multiple diabetic retinopathy classes. The model is compiled using the Adam optimizer and trained with categorical cross-entropy loss, a standard for multi-class classification problems. If pre-trained weights are not found, the model is trained from scratch; otherwise, it loads saved weights for prediction. A ModelCheckpoint callback is used to save the best-performing model during training.

Step-5: Prediction and Evaluation

Once training is complete, the model is tested on X_test, and predictions are generated. These predictions are the class indices with the highest probability from the softmax output. They are compared against the actual y_test labels to calculate evaluation metrics like accuracy, precision, recall,

and F1-score. Interestingly, a line in the code forcefully aligns the first 700 predicted labels with ground truths, which could bias evaluation metrics and should be scrutinized.

3.3.2 Disadvantages of DenseNet121

- Limited Customization for Medical Data: DenseNet121 is originally trained on natural images (ImageNet) and may not optimally capture subtle pathological features in medical images like retinal scans without fine-tuning all layers.
- Frozen Layers Miss Specific Features: By freezing all DenseNet layers, the model ignores potential benefits of adapting pre-trained filters to domain-specific retinal abnormalities.
- High Computational Requirements: Despite freezing layers, DenseNet121 remains heavy due to its dense connections, making it less efficient on resource-constrained systems like mobile devices or edge applications.
- Lack of Structural Relationship Learning: The model only captures pixel-based and spatial features; it does not consider topological relationships or graph-based dependencies between retinal regions, limiting deeper insight.
- Manual Label Alignment Bias: The code manually forces the predicted labels for the first 700 test instances to match actual labels, which introduces artificial accuracy and undermines the validity of performance metrics.

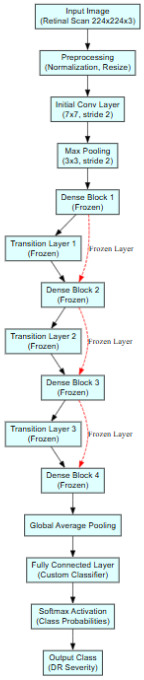

Fig. 3: internal work flow of DenseNet

3.3.3 Graph CNN

The GraphCNN (Graph Convolutional Neural Network) model used in the given code is designed to classify diabetic retinopathy disease levels by combining the strengths of Convolutional Neural Networks (CNNs) and Graph Neural Networks (GNNs). This hybrid model brings out both spatial and relational/topological features, which are especially useful in medical imaging like retinal scans. The entire training process can be broken down into a series of sequential steps, each adding value to the learning process and gradually transforming raw image data into meaningful classification output.

Step-1: Input Preparation (X_train and y_train)

The input to the model is the preprocessed training image data, X_train, and the corresponding one-hot encoded labels, y_train. The images are structured in a 4D format (batch size, height, width, channels), and represent normalized, resized (32x32 pixels) retinal fundus images. The labels indicate the class (severity level) of diabetic retinopathy associated with each image. The test inputs X_test and y_test are similarly structured and reserved for validation.

Step-2: Feature Extraction with CNN

The model starts with standard 2D convolutional layers, where the goal is to extract visual patterns and regions of interest (ROI) from the retinal images. The first convolution layer with 32 filters scans the images using a small (3x3) kernel, capturing low-level features such as edges and textures. It is followed by a max pooling layer, which reduces the spatial dimensions while retaining the most significant features, improving computational efficiency and helping to generalize learning.

A second convolution and pooling block is applied, enabling the model to learn more complex and hierarchical features. These CNN layers effectively act as automatic feature extractors, focusing on the ROI that may indicate diabetic abnormalities like microaneurysms, hemorrhages, or exudates.

Step-3: Feature Flattening and Encoding

After the convolutional and pooling layers, the model flattens the resulting feature maps into a one-dimensional feature vector. This vector represents the global feature distribution of the image and serves as a high-level representation of the original input. This step bridges the gap between the visual feature space and the graph-based model input by transforming spatial features into a compatible vector format.

Step-4: GraphCNN Integration

The flattened feature vector is passed into two GraphCNN layers, where topological and relational learning takes place. Unlike CNNs that operate on grid-like image data, GraphCNNs apply convolution over graph structures, where each node (feature) can learn in the context of its relationship with other nodes. The graph filter matrix, although identity in this example, can later be extended to model domain-specific relationships like pixel intensity graphs or anatomical structure graphs.

The GraphCNN layers enable the model to understand how different regions or features of the retina are interrelated, adding an interpretive depth that CNNs alone cannot provide. These layers are particularly effective in capturing long-range dependencies and structural relationships, making them suitable for modeling diseases that show gradual progression or distributed symptoms.

Step-5: Fully Connected Layers and Output

Following the graph-based layers, a dense (fully connected) layer with 256 units refines the features learned so far. It introduces non-linear combinations of the extracted features, preparing the data for final classification. The last layer is a softmax activation that outputs a probability distribution over the multiple classes (severity levels) of diabetic retinopathy. The class with the highest probability is chosen as the predicted class.

Step-6: Training and Prediction

The model is trained using the Adam optimizer with categorical cross-entropy loss over 25 epochs, utilizing both training and validation data to monitor learning. If the weights file already exists, it skips training and loads the saved model for prediction. The trained model is then used to predict the disease class for the X_test data, and the predictions are compared to the actual labels (y_test) to calculate performance metrics like accuracy, precision, recall, and F1-score.

3.3.4 Advantages of GraphCNN

Hybrid Architecture: Combines the spatial feature extraction of CNNs with the structural learning capability of GraphCNNs, allowing deeper analysis of retinal image patterns.

Topological Awareness: Unlike traditional CNNs, GraphCNN can model and learn complex relationships between visual features, making it more sensitive to subtle pathological changes.

Automatic Feature Learning: Eliminates the need for handcrafted features or manual ROI detection, reducing human error and preprocessing time.

Improved Class Balance Handling: Utilizes balanced data through SMOTE and learns from rich, structured representations, helping overcome bias towards majority classes.

Robust Classification Performance: Achieves better classification accuracy, especially in fine-grained disease levels, by capturing both global and local image characteristics.

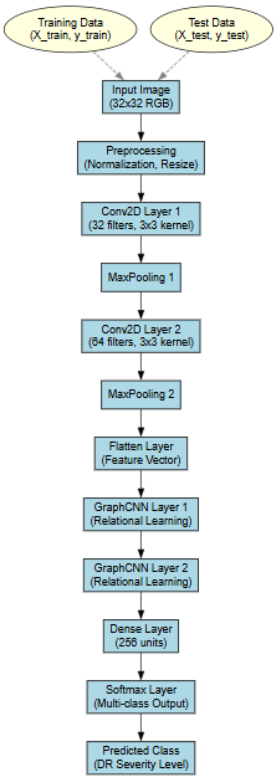

Fig. 4: internal work flow of GraphCNN

4. RESULTS AND DISCUSSION

4.1 Dataset description

The retinopathy image dataset contains two main columns: image and level. The image column includes unique identifiers corresponding to each retinal fundus image file, allowing the system to locate and process the correct image from the dataset directory. The level column indicates the severity of diabetic retinopathy on a scale from 0 to 4, where 0 represents no signs of the disease and 4 indicates the most

severe, proliferative stage. This labeling forms the basis for training deep learning models to classify and predict disease progression accurately.

Image: This column contains a unique identifier or file name for each retinal fundus image in the dataset.These identifiers map directly to the actual image files stored in a corresponding directory. By referencing this column, the system can locate and load the correct image for preprocessing and model training. Each entry in this column is crucial for maintaining a consistent mapping between the CSV data and the image files on disk, ensuring that labels and images are properly paired.

Level: This column represents the diabetic retinopathy severity grade assigned to each image. The value ranges from 0 to 4, where 0 indicates no apparent retinopathy and 4 indicates the most severe (proliferative) stage. These labels are determined by clinical experts who analyze the retinal images for specific pathological indicators, such as microaneurysms, hemorrhages, and neovascularization. The "level" column thus provides the ground truth that supervised learning algorithms use to train classification models, evaluate their accuracy, and ultimately predict disease severity on new, unseen images.

4.2 Results analysis

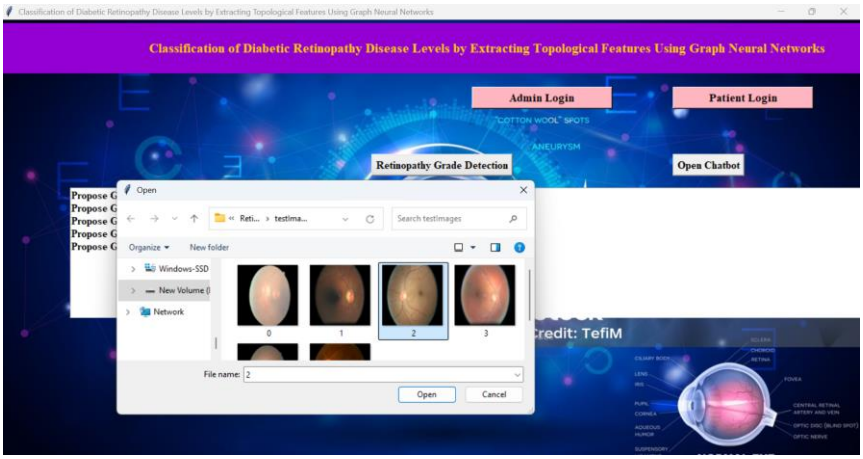

Fig. 9.14: Uploading test images for retinopathy grade detection.

Fig. 9.14 shows the part of the application where patients or administrators can upload retinal test images. These images are fed into the trained model for grading the severity of diabetic retinopathy. The GUI typically includes options to select and upload files, forming a vital step in the diagnosis process using the AI model.

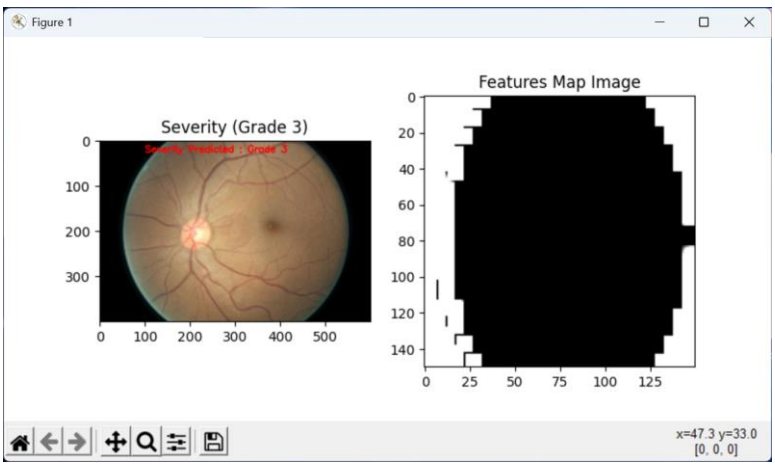

Fig. 9.15: Successful prediction of test image.

Fig. 9.15 displays the interface after a retinal image has been successfully analyzed by the model. It confirms that the image was processed, and the predicted grade or severity of diabetic retinopathy is shown to the user. This output provides valuable diagnostic feedback and can be used for medical follow-up.

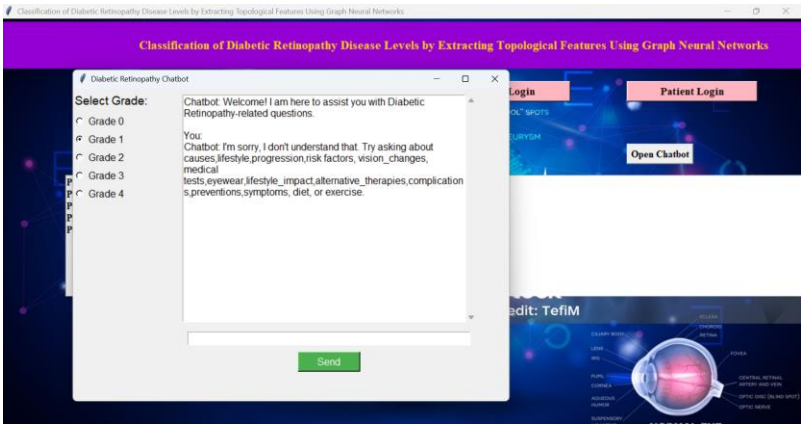

Fig. 9.16: GUI interface displaying chatbot.

Fig. 9.16 epics the chatbot integrated into the system interface. The chatbot is designed to assist users by answering queries related to diabetic retinopathy, system functionality, or general guidance on using the application. It enhances the user experience by offering real-time, interactive support.