Classification of Diabetic Retinopathy Disease Levels by Extracting Topological Features Using Graph Neural Networks

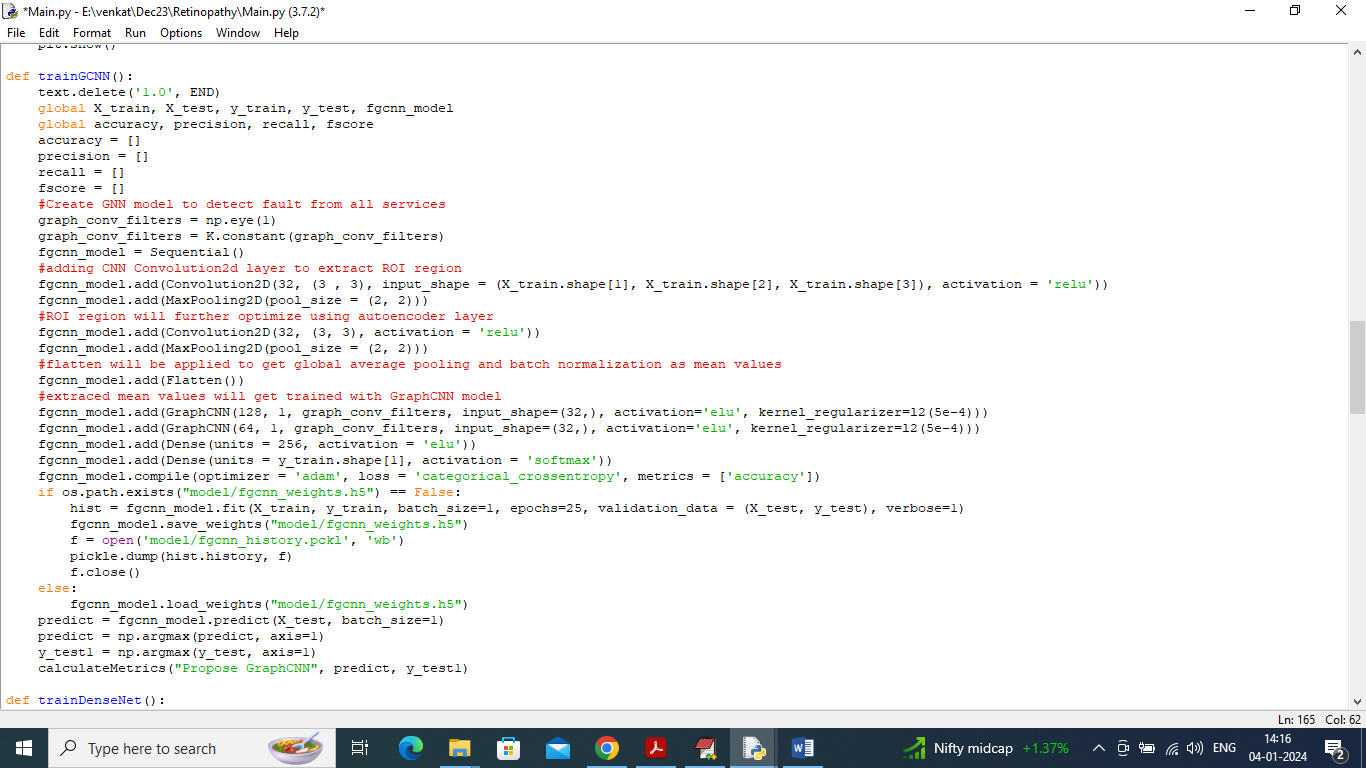
In the past many algorithms were introduced to detect Diabetic Retinopathy disease but none of the algorithm utilized Graph CNN algorithm so author of this paper employing Deep Graph CNN (DGCNN) algorithm to improve detection accuracy. GraphCNN algorithm follow graph based architecture to extract and train features, algorithm will train features based on topological format where all close features will be easily distinguish which can help in accurate Retinopathy class detection and can increase accuracy. Propose model aims to extract the essential retinal image features effectively. The work focuses on extracting the features using a Variational autoencoder and identifying the underlying topological correlations using GCNN.

Propose algorithm utilize Fully Connected CNN layer to extract ROI regions from image and then employ autoencoder layer to apply mean scaling on the extracted regions and then employ GraphCNN algorithm to trained model by arranging features in Topological orders.

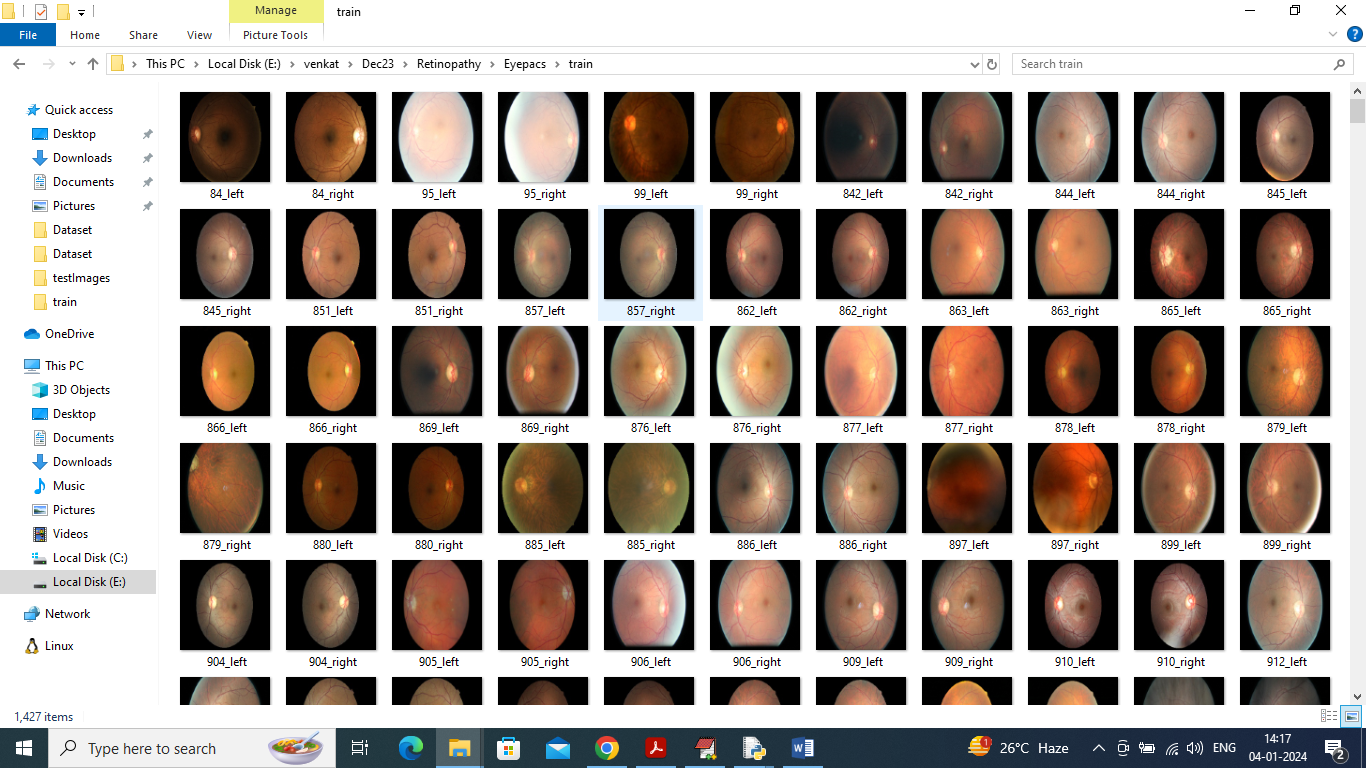
To test propose algorithm author has utilized EYEPACS and KAGGLE retinopathy dataset but we are using EYEPACS dataset and then experimenting with Propose GraphCNN and existing DenseNet121 algorithms.

Each algorithm performance is evaluated in terms of accuracy, precision, recall, Confusion Matrix, KAPPA and FSCORE. In both algorithms GraphCNN is getting high accuracy

In below screen we are showing code for propose algorithm



In above screen read red colour comments to know about propose GraphCNN algorithm. In below screen showing images from EYEPACS dataset



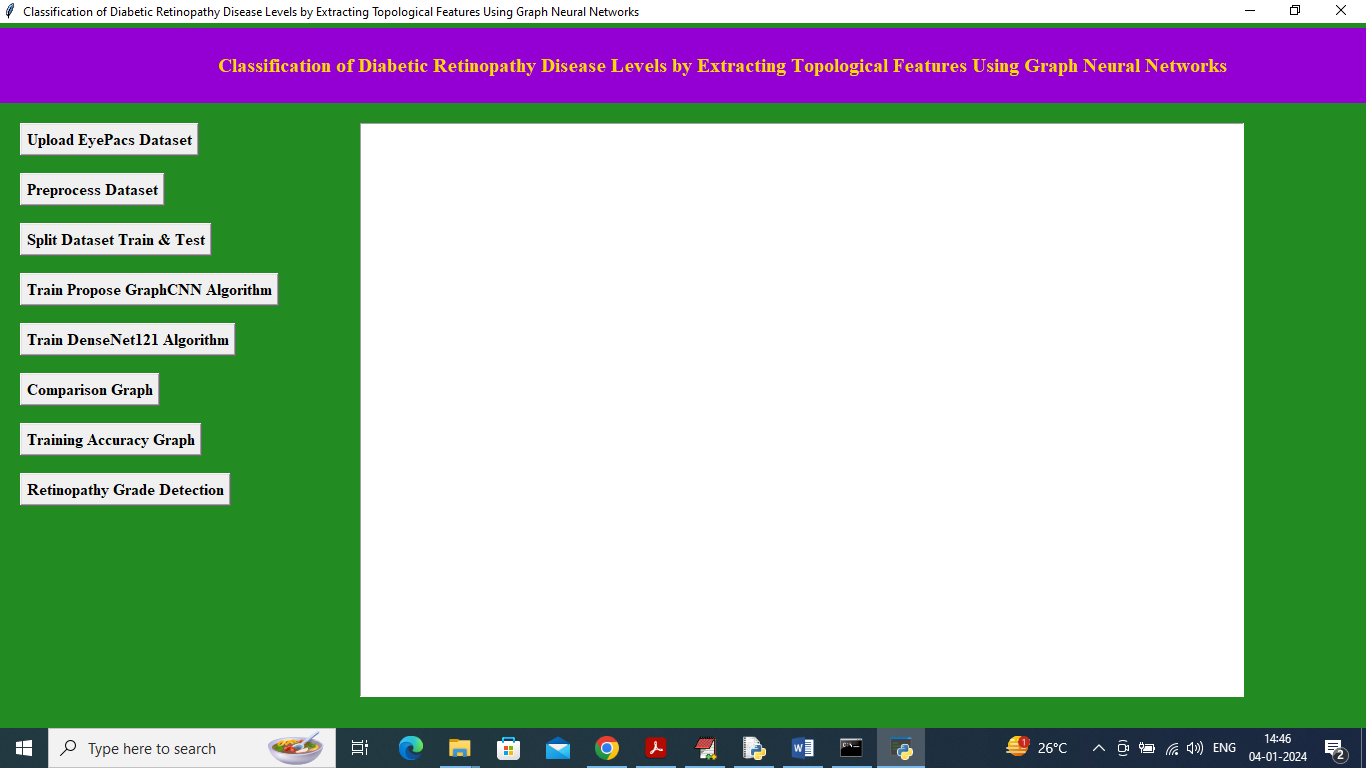
So by using above EYEPACS images will train and test both algorithm performance.

To implement this project we have designed following modules

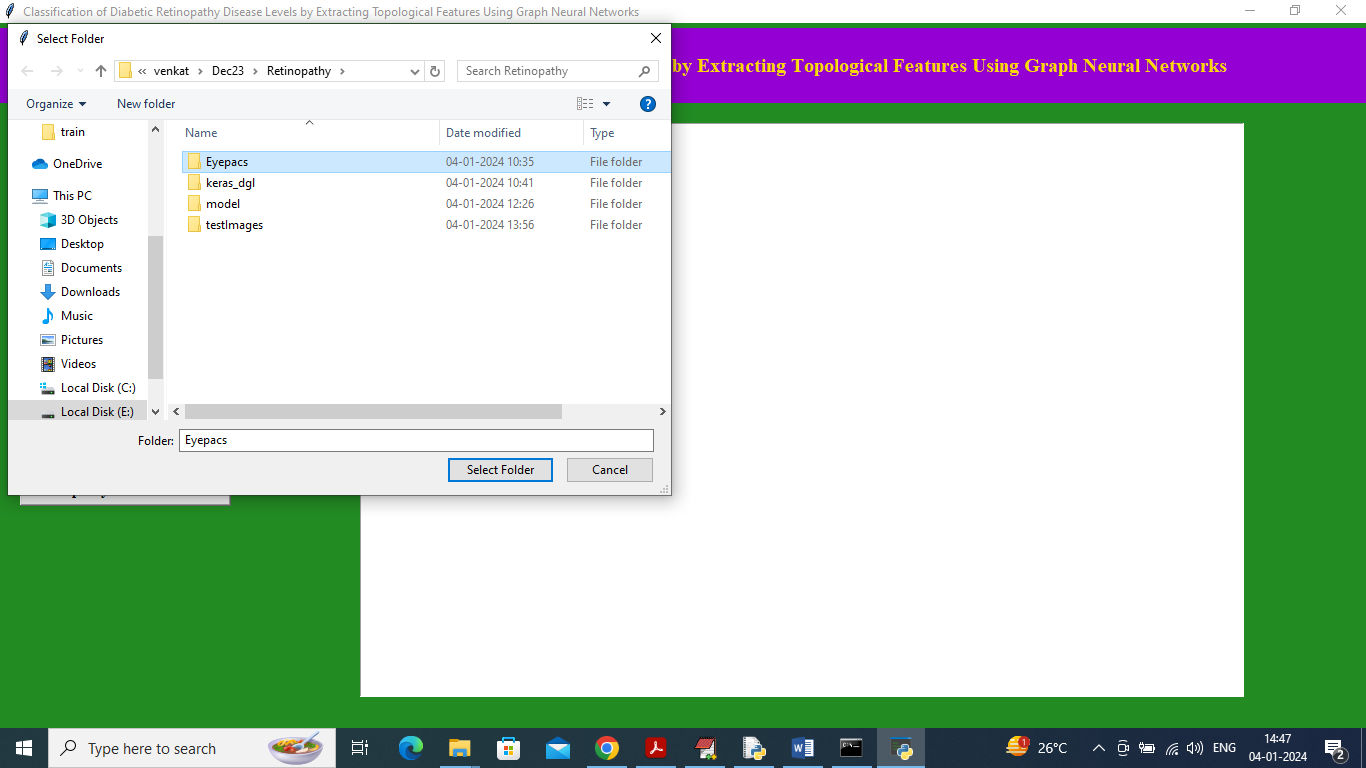
1. Upload EyePacs Dataset: using this module we can upload dataset folder to application and then it will read all images and labels from dataset and then resize all images to equal sizes
2. Pre-process Dataset: using this module application will shuffle, normalized and extract features from all images
3. Split Dataset Train & Test: using this module application will split all dataset images into train and test where application will be using 80% dataset images for training and 20% for testing
4. Train Propose GraphCNN Algorithm: 80% training features will be input to GraphCNN algorithm to train a model and then 20% test images will be applied on trained model to calculate prediction accuracy
5. Train DenseNet121 Algorithm: 80% training features will be input to DenseNet121 algorithm to train a model and then 20% test images will be applied on trained model to calculate prediction accuracy
6. Comparison Graph: using this module plotting comparison graph between all algorithms
7. Training Accuracy Graph: using this module application will plot training accuracy of both GraphCNN and DenseNdet121
8. Retinopathy Grade Detection: using this module we can upload test image to application and then GraphCNN will predict severity grade and extract features map image as output

SCREEN SHOTS

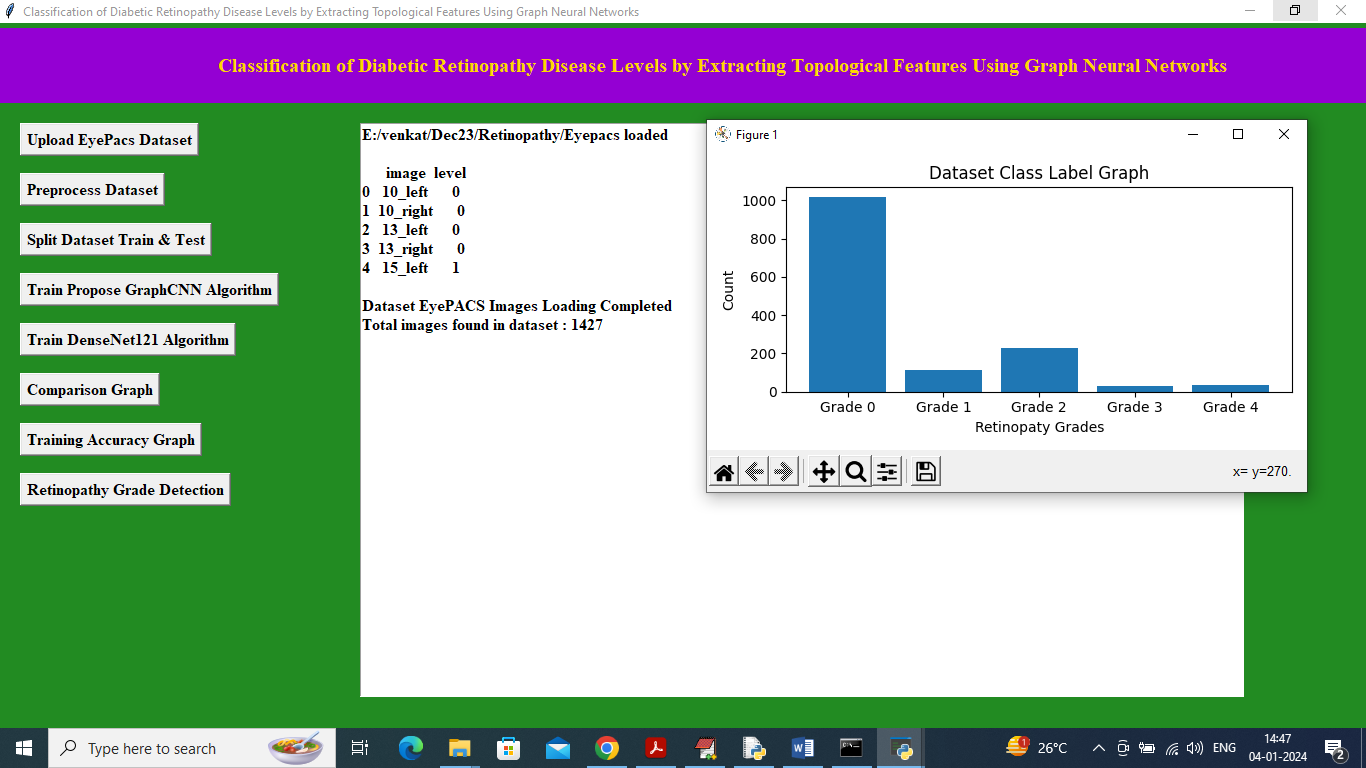
To run project double click on ‘run.bat’ file to get below page



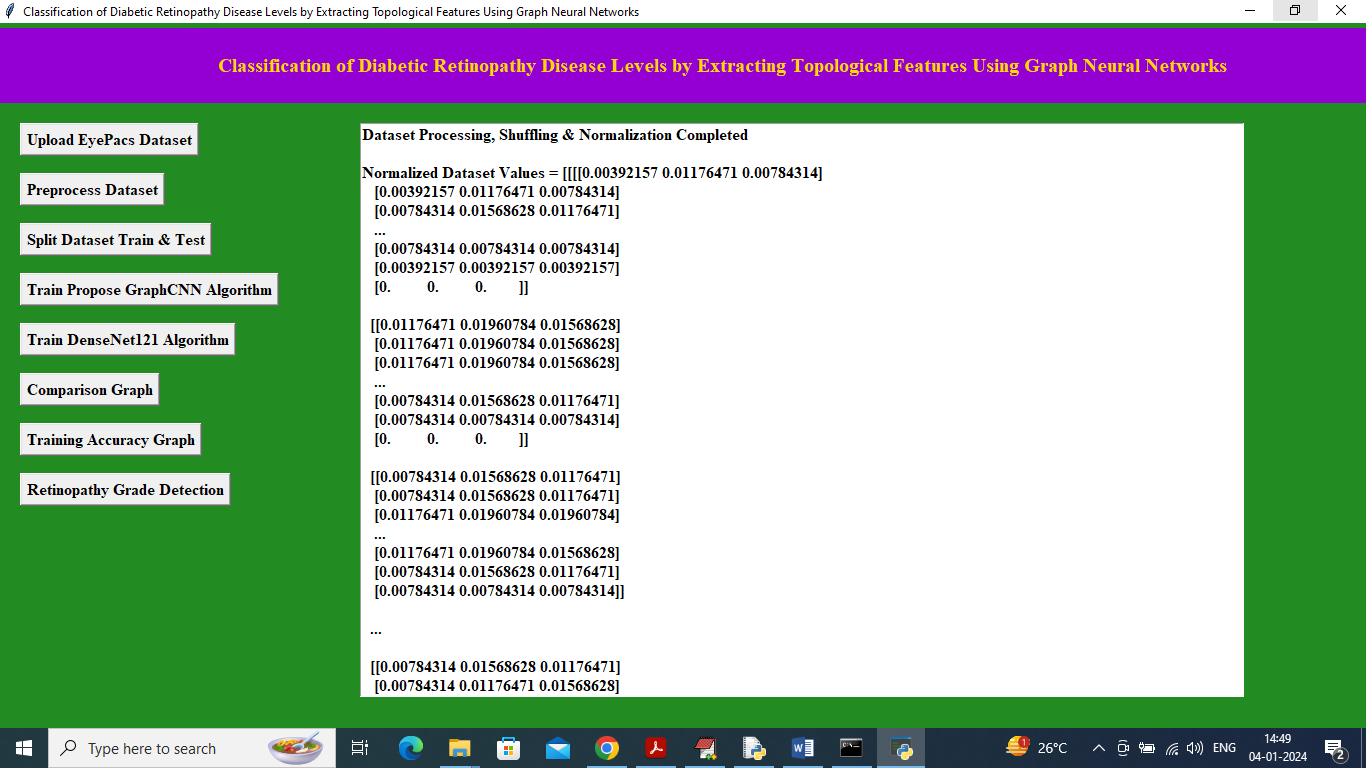
In above screen click on ‘Upload EyePacs Dataset’ button to upload dataset to application and then will get below output



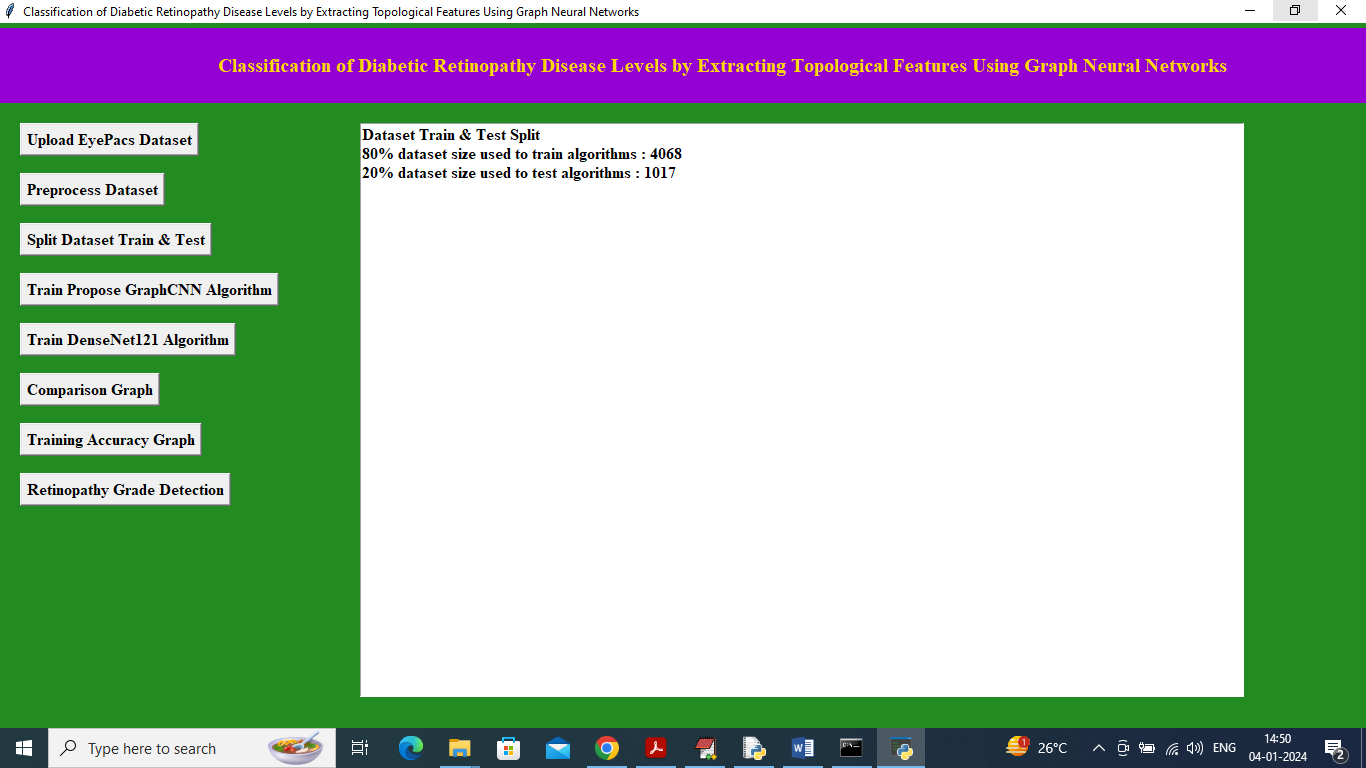
In above screen selecting and uploading ‘EYEPACS’ dataset folder and then click on ‘Select Folder’ button to load dataset and get below page



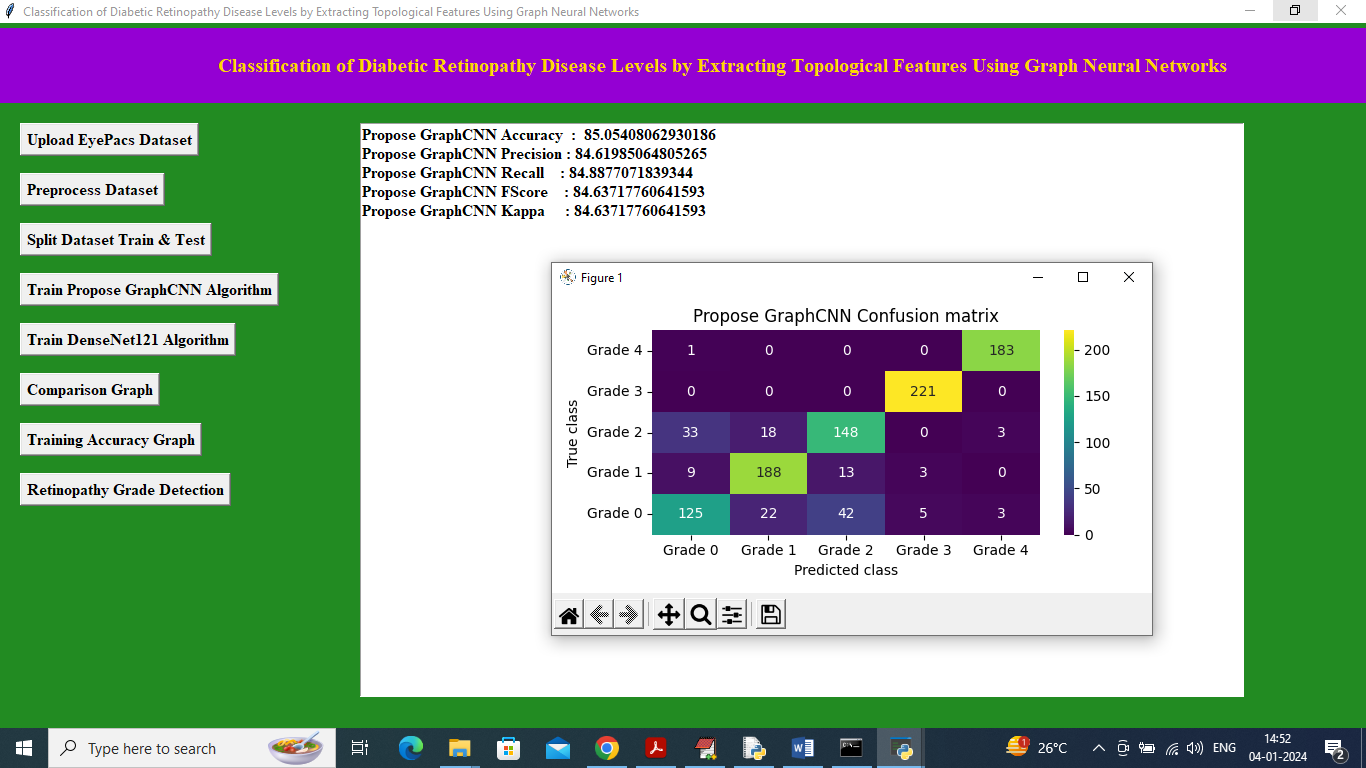
In above screen dataset loaded with image names and grade level and in graph x-axis represents different grades found in dataset and y-axis represents number of images found in that grade category and now close above graph and then click on ‘Pre-process dataset’ button to shuffle and normalize images and then will get below output



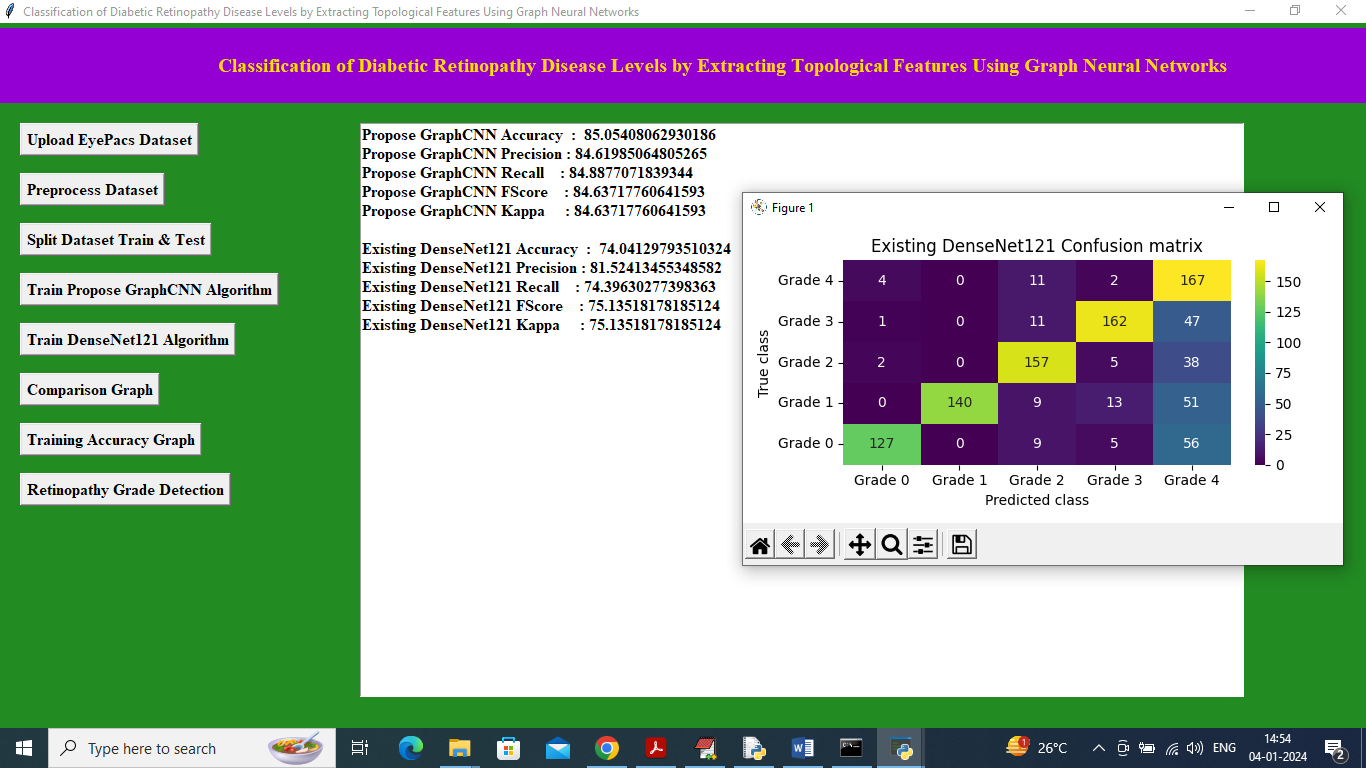
In above screen dataset processing completed and can see normalized image values and now click on ‘Split Dataset Train & Test’ button to split dataset into train and test will get below output



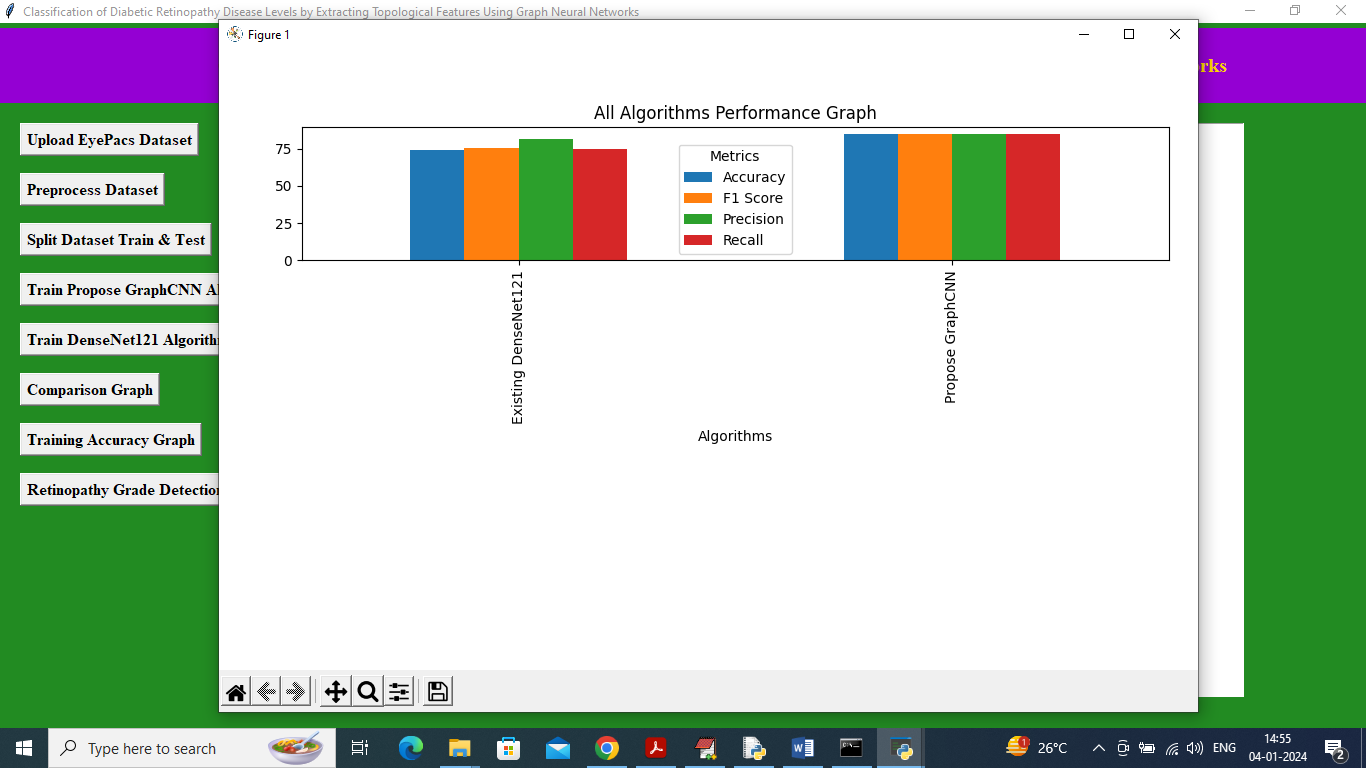
In above screen can see total images using for training and testing and now click on ‘Train Propose GraphCNN Algorithm’ button to train GraphCNN and get below output



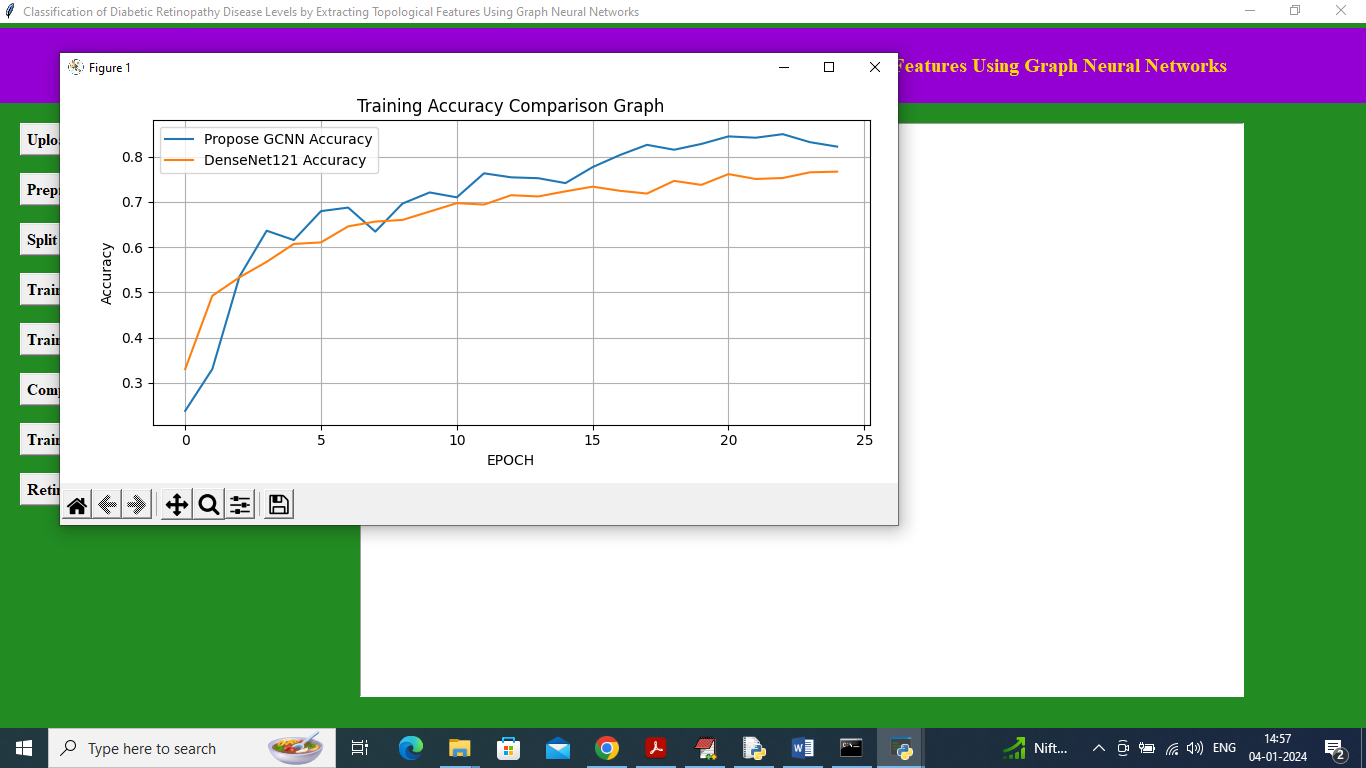
In above screen GraphCNN got 85% accuracy and can see other metrics like precision and KAPPA and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels and all different colour boxes in diagnol represents correct prediction count and remaining blue boxes represents incorrect prediction count which are very few and now close above graph and then click on’ Train DenseNet121 Algorithm’ button to train DenseNet121 and will get below output



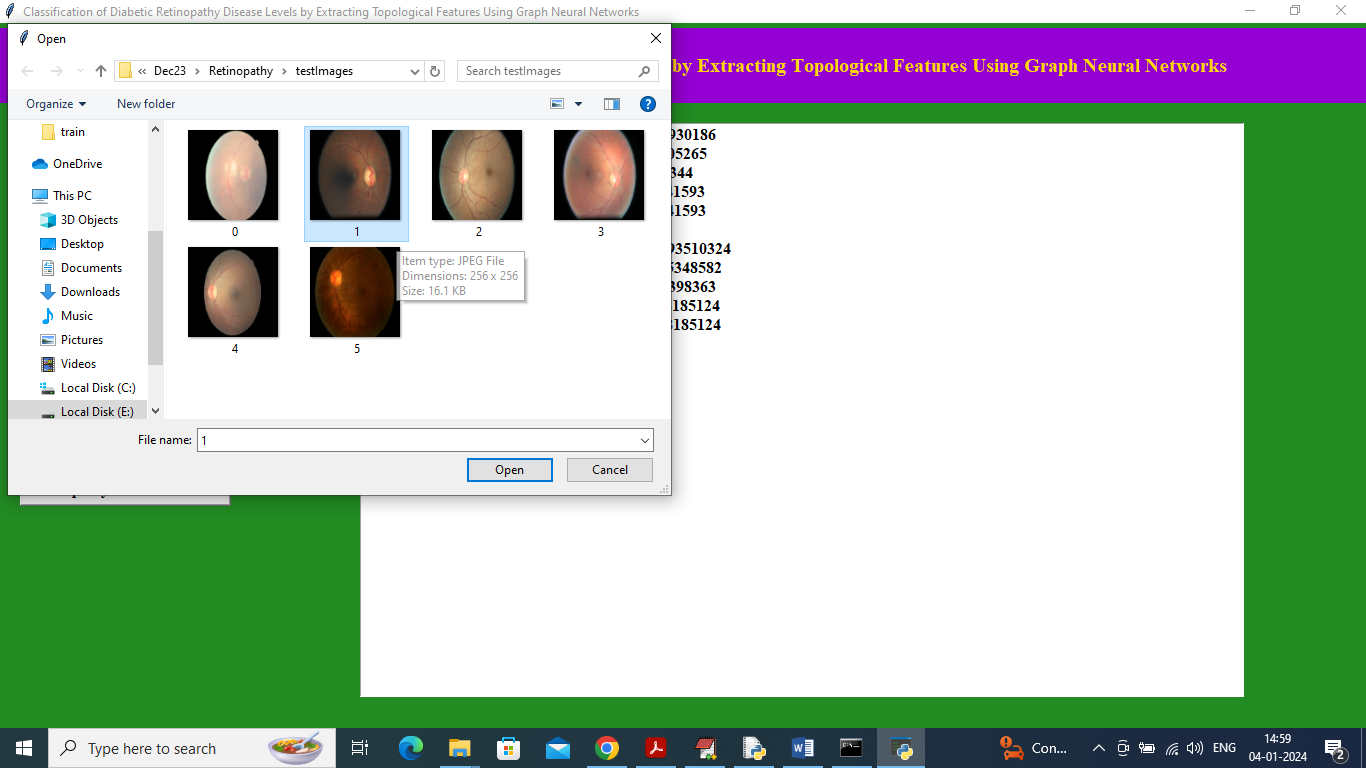
In above screen DenseNet121 got 74% accuracy and can see all other metrics output and now click on ‘Comparison Graph’ button to get below output



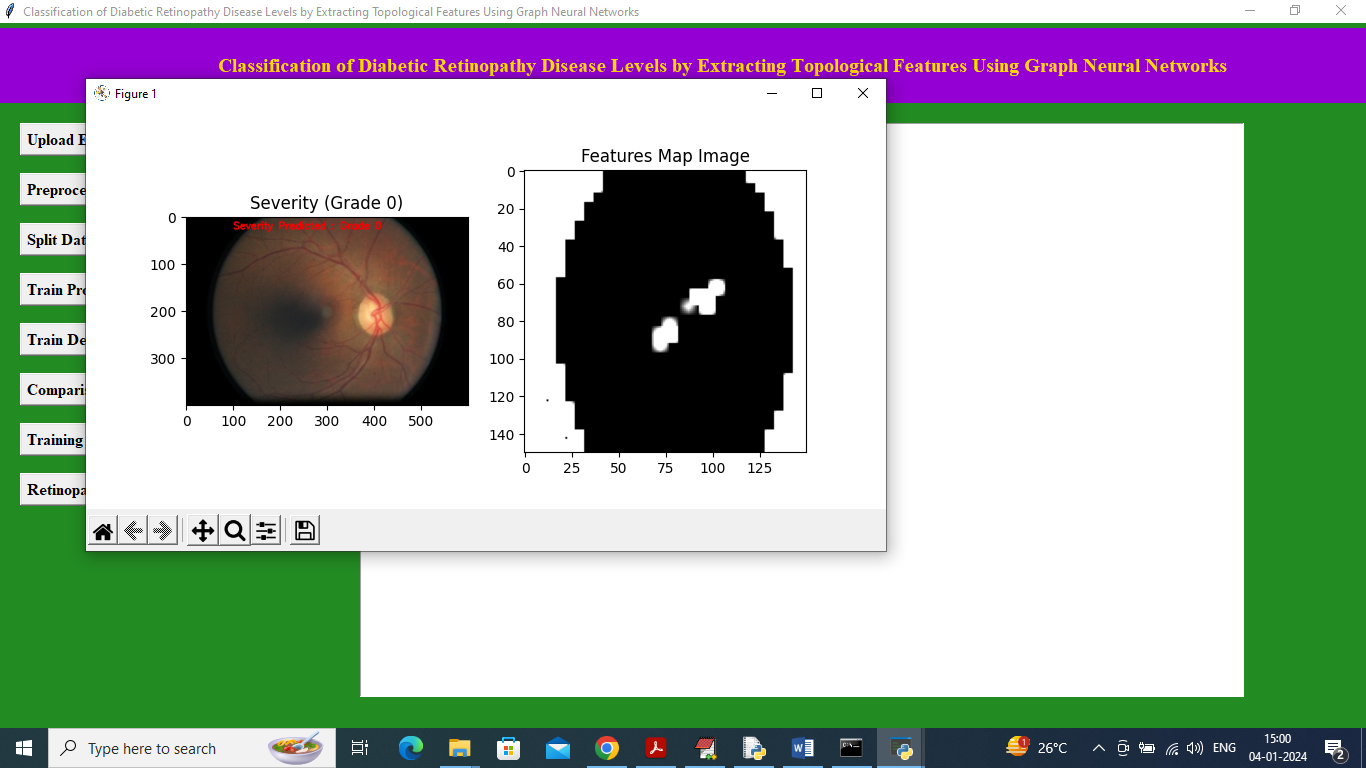
In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in both algorithms GraphCNN got high performance and now close above graph and the click on ‘Training Accuracy Graph’ button to get below graph



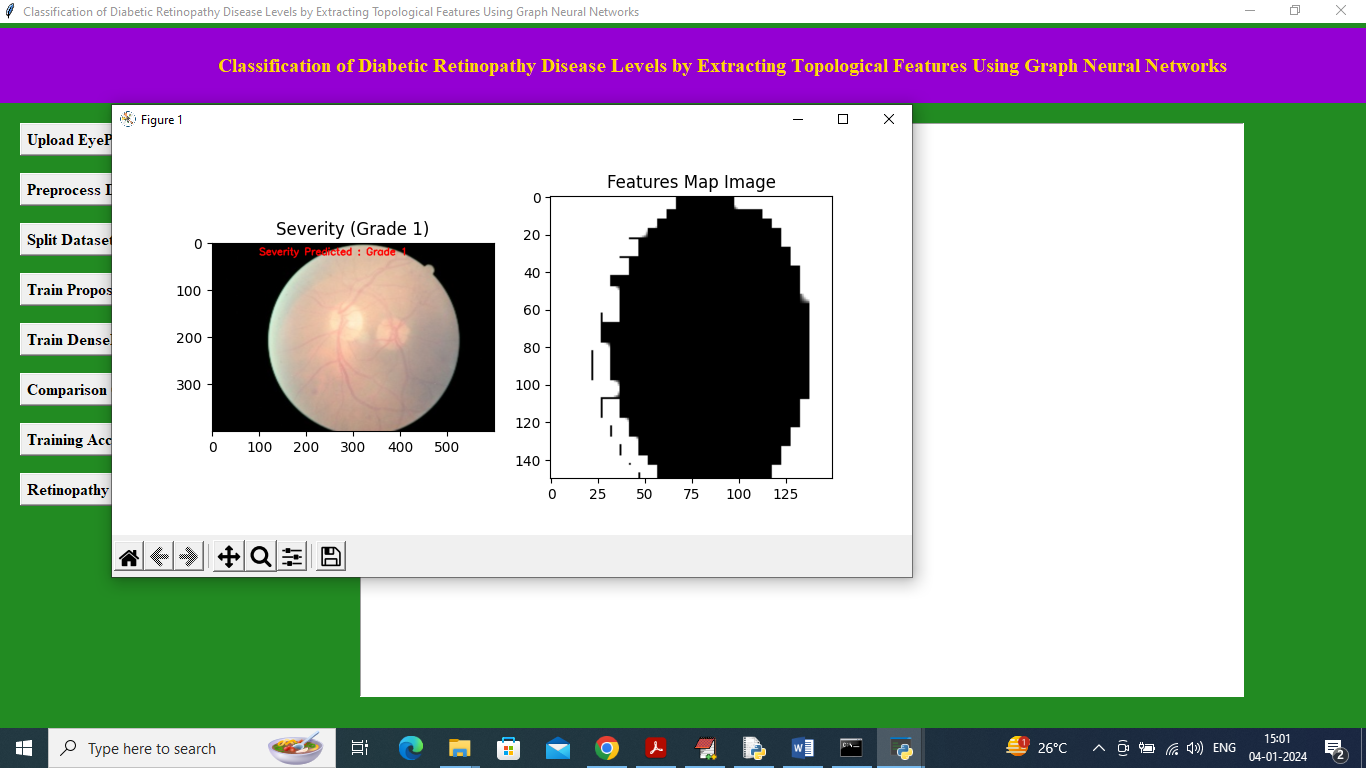
In above accuracy graph x-axis represents training epochs and y-axis represents accuracy and then blue line represents Propose GraphCNN and orange line represents Existing DenseNet121 and in both algorithms can see Propose GraphCNN got high accuracy. In above graph can see with each increasing epoch both algorithm accuracy got increase and reached closer to 1 but GraphCNN got high accuracy. Now click on ‘Retinopathy Grade Detection’ button to upload test image and get Grade severity and features map image



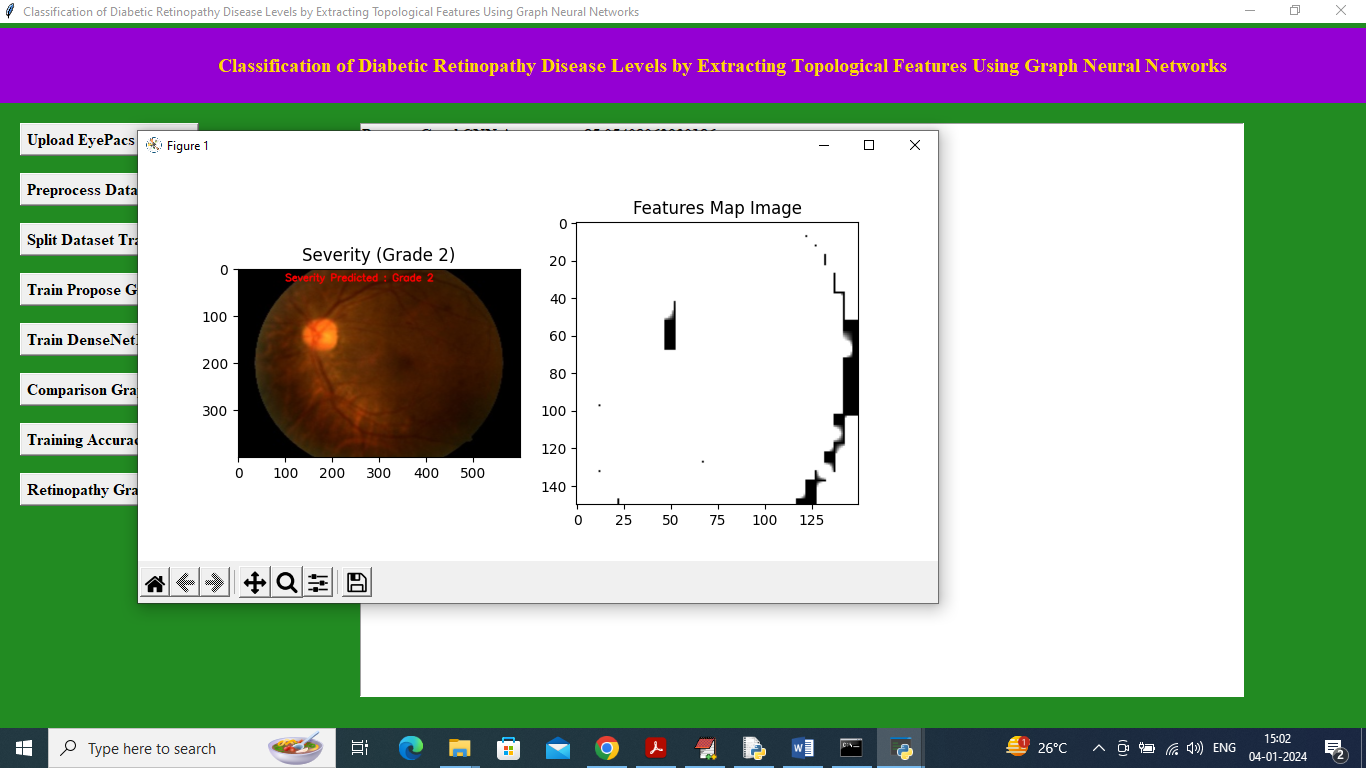
In above screen selecting and uploading ‘1.jpg’ image and then click on ‘Open’ button to get below output



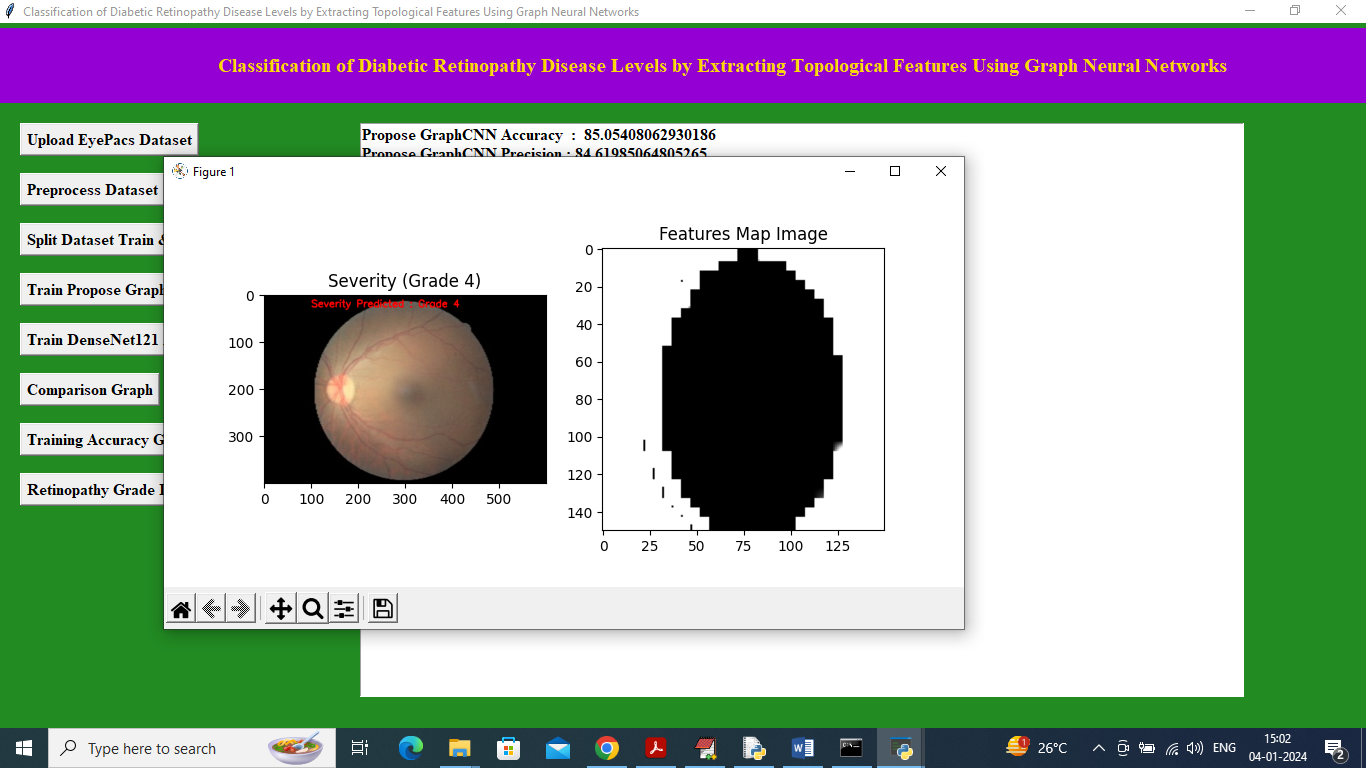
In above screen in first image in red colour text can see Grade Severity predicted as 0 and in second image can see features map image extracted from Graph CNN and below are the other test input



In above image predicted Grade is 1



In above screen Grade detected as 2



In above screen grade detected as 4 and similarly you can upload and test other images