**ABSTRACT**

The eye is sometimes said to provide a window into the health of a person, for it is only in the eye that one can actually see the exposed flesh of the subject without using invasive procedures. Diabetic retinopathy (DR) is a serious eye disease originating from diabetes mellitus. There are a number of diseases, particularly vascular disease, that leave tell-tale markers in the retina. Microaneurysms (MAs) are early signs of DR, so the detection of these lesions is essential in an efficient screening program to meet clinical protocols. Retinal images provide considerable information on pathological changes caused by local ocular disease which reveals diabetes, hypertension, arteriosclerosis, cardiovascular disease, and stroke. Computer-aided analysis of retinal images plays a central role in diagnostic procedures. However, automatic retinal segmentation is complicated by the fact that retinal images are often noisy, poorly contrasted, and the vessel widths can vary from very large to very small. This project presents image processing techniques such as dark object detection to analyse the condition or enhance the input image in order to make it suitable for further processing and improve the visibility of vessels in color fundus images. Then we can implement an automated classification algorithm named as Convolutional Neural Network (CNN) algorithm, specifically using the VGG16 architecture. The CNN architecture is designed to effectively extract features from retinal images, capturing intricate patterns associated with diabetic retinopathy. The model is trained using a combination of loss functions and optimization techniques to ensure convergence and generalization. Hyperparameter tuning is performed to optimize the model's performance on the validation set. The trained CNN is evaluated on a separate test set, and its performance metrics, including accuracy, precision, recall, and F1 score, are reported. Additionally, the model's interpretability is explored to understand the features contributing to predictions.

**CHAPTER 1**

**INTRODUCTION**

**1.1 DIABETIC RETINAL DISEASE PREDICTION**

Diabetic Retinopathy (DR) is one of the most critical complications associated with diabetes mellitus and stands as a leading cause of vision impairment and blindness globally. The retina, being a sensitive tissue at the back of the eye, plays a pivotal role in vision by converting light into neural signals. In diabetic patients, prolonged exposure to high blood glucose levels damages the retinal blood vessels, leading to complications such as microaneurysms, hemorrhages, exudates, and in severe cases, retinal detachment. Given the global rise in diabetes cases, early detection and diagnosis of DR are essential to prevent irreversible vision loss. Traditionally, the screening of DR relies on manual examination of retinal images by ophthalmologists, a process that is time-intensive, subjective, and not scalable, especially in rural or underserved regions. With the increase in diabetic population, this traditional approach is becoming less viable, highlighting the urgent need for automated diagnostic tools. Advanced technologies like deep learning and computer vision offer promising alternatives for the early detection and classification of DR. In particular, Convolutional Neural Networks (CNNs) have demonstrated exceptional performance in image analysis and medical diagnosis tasks. This project proposes an AI-driven solution utilizing the VGG16 CNN architecture to analyze retinal fundus images for signs of diabetic retinopathy and glaucoma. The system follows a structured pipeline involving image acquisition, preprocessing, segmentation, feature extraction, and classification. Through training on labeled retinal datasets, the model learns to detect key abnormalities indicative of DR. The system is evaluated using accuracy, precision, recall, and F1-score metrics, ensuring reliable performance. By leveraging the deep learning capabilities of VGG16, this approach enhances diagnostic accuracy, reduces the burden on healthcare professionals, and facilitates early medical intervention.

**1.1.1 DIABETIC RETINA AND ITS CLINICAL SIGNIFICANCE**

Diabetic Retinopathy (DR) is a progressive eye disease that arises as a complication of long-term diabetes mellitus. The retina, which is responsible for capturing visual signals, is highly sensitive to changes in blood glucose levels. In diabetic individuals, high glucose concentrations damage the small blood vessels in the retina, leading to leakage, hemorrhages, and the formation of abnormal blood vessels. This can result in vision distortion and, if left untreated, permanent blindness. Clinical signs of DR include microaneurysms, hard and soft exudates, neovascularisation, and retinal detachment, which are critical indicators of disease severity. Understanding and identifying these pathological features in retinal images is essential for timely diagnosis and treatment. Manual examination by ophthalmologists is accurate but time-consuming and impractical for mass screening, especially in rural or resource-limited settings. Hence, there is a significant need for automated systems that can detect DR features at an early stage. Such systems must be robust enough to handle image noise, variations in lighting, and differences in vessel width, all of which commonly affect retinal imaging quality. This forms the clinical motivation behind developing AI-based DR detection systems.

**1.1.2 DIABETIC RETINA CLASSIFICATION USING CNN AND VGG16**

Convolutional Neural Networks (CNNs) are a class of deep learning models particularly well-suited for image classification tasks. The VGG16 architecture, known for its deep layers and consistent convolutional structure, has proven highly effective in extracting detailed visual features. In this project, VGG16 is utilized to automatically classify retinal images into categories indicating the presence or absence of diabetic retinopathy, and potentially different stages of the disease. The model is trained using labeled datasets where each retinal image is annotated with clinical information regarding DR presence and severity. The process begins with image preprocessing to enhance visual quality applying techniques like dark object detection, contrast normalization, and noise filtering. After preprocessing, the images are passed through the VGG16 model, where the network’s multiple convolutional layers extract complex hierarchical features, such as texture patterns and vessel morphology. These features are then used by the classification layers to determine the disease class. Hyperparameter tuning and validation help improve the model’s generalization ability, ensuring it performs well on unseen data. The model is evaluated using metrics like precision, recall, F1-score, and accuracy, providing a comprehensive view of its diagnostic reliability.

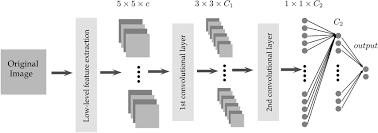
**1.1.3 AUTOMATED DIAGNOSIS AND CLINICAL DEPLOYMENT**

The final stage of the project focuses on deploying the trained model in a clinical or screening environment to support ophthalmologists and primary care physicians. Once trained and validated, the model can automatically process new retinal images and provide immediate diagnostic feedback. The system can highlight critical regions of interest (ROIs) such as microaneurysms or hemorrhages using activation or saliency maps, increasing transparency and interpretability for medical professionals. These visualizations allow doctors to understand the reasoning behind the model’s predictions and build trust in AI-driven diagnostics. Additionally, the system’s architecture allows for real-time implementation, making it suitable for integration into telemedicine platforms or community healthcare systems. It can significantly reduce the burden on ophthalmologists by pre-screening patients and flagging only those with likely DR for further manual review. Furthermore, expanding the system’s capability to detect other eye conditions such as glaucoma strengthens its utility. Overall, this project contributes to the growing field of AI in healthcare by providing an accurate, scalable, and interpretable tool for diabetic eye disease diagnosis, ultimately supporting early treatment and improving patient outcomes.

**1.2 DEEP LEARNING**

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behaviour of the human brain albeit far from matching its ability allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars). Deep learning is a subfield of machine learning that uses artificial neural networks to model and solve complex problems. It has emerged as one of the most promising areas of research in artificial intelligence and has been applied to a wide range of applications such as image and speech recognition, natural language processing, and robotics. Deep learning models are based on artificial neural networks that are inspired by the structure and function of the human brain. These networks consist of layers of interconnected nodes, each of which performs a mathematical operation on the input data. The output of each node is passed on to the next layer of nodes, where it is combined with the outputs of other nodes and further processed. This process continues until the output of the final layer is produced, which represents the prediction or classification of the input data.

One of the key advantages of deep learning is its ability to learn complex patterns and relationships in the data. This is achieved by using multiple layers of nodes, each of which learns a different set of features from the input data. The first layer learns low-level features such as edges and corners, while subsequent layers learn higher-level features such as textures and shapes. This hierarchical learning process enables deep learning models to capture complex patterns and relationships in the data, making them highly effective in solving complex problems. Another advantage of deep learning is its ability to learn from large amounts of data. Deep learning models require large amounts of data to train effectively, but once trained, they can make accurate predictions on new, unseen data. This makes deep learning particularly well-suited for applications such as image and speech recognition, where large amounts of labeled data are available. Deep learning has also benefited from the availability of powerful hardware such as GPUs and TPUs, which can accelerate the training and inference of deep learning models. This has enabled researchers and developers to train larger and more complex models, leading to significant improvements in performance and accuracy. Despite its many advantages, deep learning also has some limitations and challenges. One of the main challenges is the need for large amounts of labeled data. Deep learning models require labeled data to learn from, which can be difficult and expensive to obtain, especially for niche applications. Another challenge is the interpretability of deep learning models. Deep learning models are often seen as black boxes, making it difficult to understand how they arrive at their predictions or classifications. This can be problematic in applications where interpretability is important, such as in healthcare or finance. In conclusion, deep learning has emerged as a powerful and versatile tool for solving complex problems in a wide range of applications. Its ability to learn complex patterns and relationships in the data, and its scalability to large datasets, make it particularly well-suited for applications such as image and speech recognition. However, the need for large amounts of labeled data and the interpretability of deep learning models are still challenges that need to be addressed. As deep learning continues to evolve, it is likely that these challenges will be overcome, leading to even more sophisticated and accurate models.



**Figure 1.1 Deep learning**

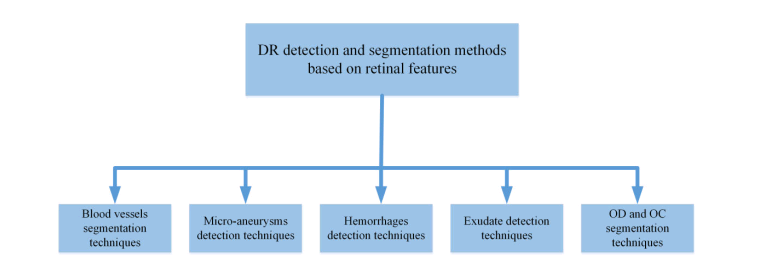
**1.3 TYPES OF DEEP LEARNING**

* **Convolutional Neural Networks (CNNs):** CNNs are primarily used for processing image data. They apply convolutional layers to detect features such as edges, textures, and shapes in images. CNNs are widely used in tasks like image classification, object detection, and facial recognition due to their ability to learn spatial hierarchies.
* **Recurrent Neural Networks (RNNs):** RNNs are designed for sequential data such as time series or natural language. They use loops within the network to retain memory of previous inputs, making them effective for language modeling, speech recognition, and text generation. However, they suffer from vanishing gradient problems over long sequences.
* **Long Short-Term Memory Networks (LSTMs):** LSTMs are a special type of RNN that can learn long-term dependencies using memory cells and gating mechanisms. They are ideal for tasks involving long sequences like language translation, handwriting recognition, and video analysis. LSTMs overcome the limitations of traditional RNNs in handling long-term memory.
* **Gated Recurrent Units (GRUs):** GRUs are a simpler and faster alternative to LSTMs. They use gating units to control the flow of information without separate memory cells, allowing efficient learning on sequential data. GRUs are effective in tasks like sentiment analysis, time series prediction, and chatbot development.
* **Autoencoders:** Autoencoders are unsupervised neural networks used for data compression and reconstruction. They consist of an encoder that compresses the input and a decoder that reconstructs it. Autoencoders are used in anomaly detection, denoising, and dimensionality reduction.
* **Generative Adversarial Networks (GANs):** GANs consist of two networks a generator and a discriminator that compete with each other. The generator creates fake data while the discriminator tries to distinguish between real and fake data. GANs are used in generating realistic images, videos, and deepfakes.
* **Deep Belief Networks (DBNs):** DBNs are composed of multiple layers of Restricted Boltzmann Machines (RBMs). They are generative models used for unsupervised learning, feature extraction, and pre-training deep networks. Though less common today, DBNs laid the groundwork for modern deep learning.
* **Transformers:** Transformers are advanced deep learning models based on self-attention mechanisms, capable of handling long-range dependencies. They have revolutionized natural language processing tasks such as machine translation, text summarization, and question answering. BERT and GPT are popular transformer models.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Automatic Detection of Diabetic Retinopathy: A Review on Datasets, Methods, and Evaluation Metrics**

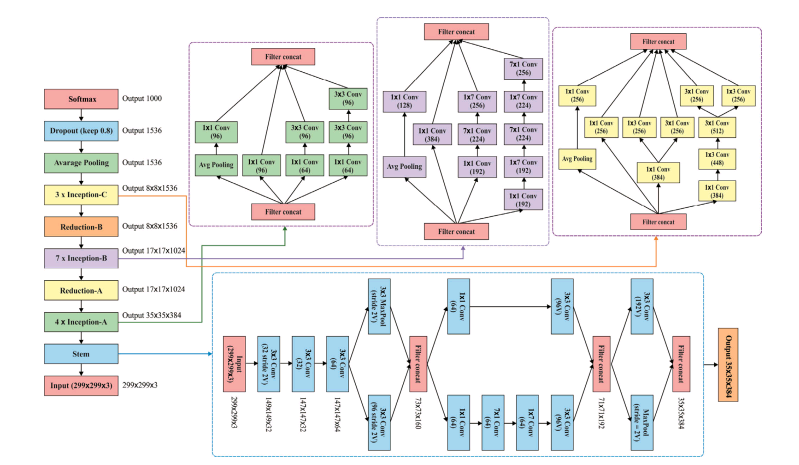
****

This paper provides a comprehensive review of automated diabetic retinopathy (DR) detection, a critical aspect of diabetes management. The authors analyze various publicly available datasets for DR classification, such as the Kaggle Diabetic Retinopathy Detection dataset and the Messidor dataset, and evaluate the strengths and limitations of each. The study highlights how these datasets facilitate the development of machine learning models that can automatically detect DR in retinal fundus images, making early detection more accessible and effective.

In terms of methods, the review covers a broad range of algorithms, from traditional machine learning techniques like support vector machines (SVMs) and random forests to more modern deep learning approaches such as Convolutional Neural Networks (CNNs). These methods are assessed for their performance in classifying DR into different stages of severity, from mild to proliferative DR. A key challenge highlighted in the review is the imbalance between healthy and affected images in many datasets, which can lead to skewed results and affect model performance. Therefore, the review suggests several strategies to address this, such as data augmentation and class weighting.

The paper also provides an overview of the most commonly used evaluation metrics, including accuracy, sensitivity, specificity, and the area under the receiver operating characteristic (ROC) curve (AUC). It emphasizes the importance of using multiple metrics to assess model performance comprehensively. Finally, the authors discuss future research directions, including the integration of multimodal data (such as demographic information) and the need for real-time, automated diagnostic systems for clinical settings.

**2.2 Hyperparameter Tuning Deep Learning for Diabetic Retinopathy Fundus Image Classification**

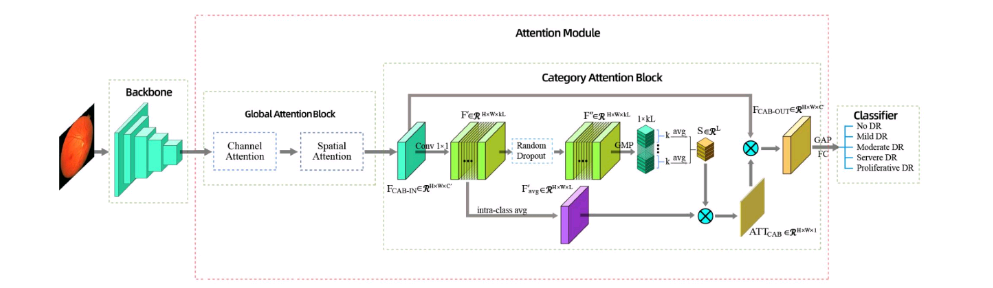
****

This research focuses on the critical role of hyperparameter tuning in improving the performance of deep learning models for diabetic retinopathy detection. The paper discusses various hyperparameters such as learning rate, batch size, and the number of layers, and how these affect the performance of CNN-based models on fundus images. Hyperparameter tuning is often an overlooked but crucial step in achieving optimal model performance, especially when working with deep learning models on complex medical imaging tasks like DR classification.

The authors explore several strategies for hyperparameter optimization, including grid search, random search, and more advanced methods like Bayesian optimization. The study presents empirical results from a series of experiments on the Kaggle Diabetic Retinopathy Detection dataset, showcasing the significant improvements that can be achieved through careful tuning of hyperparameters. For example, the authors demonstrate how adjustments to the learning rate and batch size can lead to faster convergence and higher accuracy in detecting different stages of DR.

The paper emphasizes the need for automated and systematic approaches to hyperparameter tuning in the medical imaging domain. The authors suggest that these methods can be applied to large-scale datasets and real-time clinical systems to enhance the efficiency of diabetic retinopathy diagnosis. Furthermore, the paper also touches upon the importance of interpretability in deep learning models, ensuring that clinicians can understand and trust the results provided by automated systems.

**2.3 CABNet: Category Attention Block for Imbalanced Diabetic Retinopathy Grading**

****

CABNet introduces a novel approach to handling the class imbalance issue in diabetic retinopathy grading using a Category Attention Block (CAB). In diabetic retinopathy classification, the number of healthy samples often outweighs the number of DR-affected samples, which can lead to poor model performance. The proposed CABNet integrates an attention mechanism into the CNN architecture to focus more on the minority classes, particularly the more severe stages of DR, which are typically underrepresented in datasets.

The CAB module works by assigning attention weights to different categories based on their importance, effectively addressing the imbalance between healthy and DR images. The architecture also includes a category-specific loss function that penalizes misclassifications of minority classes more heavily, further improving performance on rare but critical DR stages. The authors demonstrate how CABNet outperforms traditional CNN models in terms of both classification accuracy and recall, particularly for the detection of severe DR cases.

Experiments on several publicly available datasets, including the Messidor and EyePACS datasets, show that CABNet achieves state-of-the-art results in DR classification, even with imbalanced data. The authors also highlight that this method can be generalized to other medical imaging tasks where class imbalance is a common issue, suggesting its broader applicability in the field of medical image analysis.

**2.4 Diabetic Retinopathy Diagnosis from Fundus Images Using Stacked Generalization of Deep Models**

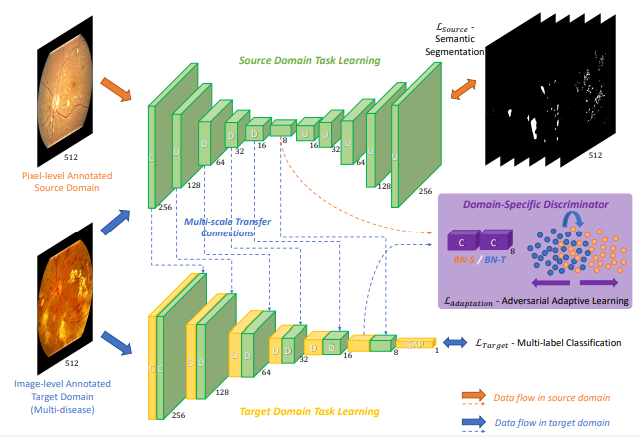
****

This paper presents a stacked generalization (or stacking) approach for diabetic retinopathy diagnosis, combining multiple deep learning models to improve classification performance. The idea behind stacking is to leverage the strengths of different models to create a more robust ensemble model. The authors apply this technique to fundus images, using several base models, such as CNNs, ResNets, and DenseNets, and then combine their predictions through a meta-model, typically a simpler model like a logistic regression or SVM.

The study demonstrates how stacking helps overcome the limitations of individual models, particularly in complex tasks like diabetic retinopathy detection, where subtle differences in image features need to be captured. By aggregating the outputs of multiple models, the stacked generalization approach reduces the risk of overfitting and increases the generalization ability of the model. The authors also investigate how different architectures and combinations of base models contribute to the overall performance, with CNN-based models showing significant improvements in feature extraction from fundus images.

Through extensive evaluation on the EyePACS and Kaggle Diabetic Retinopathy Detection datasets, the proposed model outperforms traditional CNN approaches in terms of accuracy, sensitivity, and F1-score. The results demonstrate the effectiveness of stacked generalization in improving diabetic retinopathy classification and emphasize the potential of ensemble learning techniques in the medical imaging field.

**2.5 A Benchmark for Studying Diabetic Retinopathy: Segmentation, Grading, and Transferability**

****

This paper presents a benchmark study focused on diabetic retinopathy segmentation, grading, and the transferability of models across different datasets. The authors highlight the challenges associated with DR segmentation, which is the task of identifying the exact boundaries of lesions and exudates in fundus images. The paper also provides a detailed analysis of different DR grading systems, which classify the severity of DR based on the extent of damage to the retina.

One key contribution of this work is the creation of a unified framework for evaluating DR detection models, considering not only classification accuracy but also the ability to generalize across various datasets. The study introduces a set of standardized metrics for evaluating segmentation performance and grading consistency, which helps to benchmark different models and approaches. The authors also discuss the impact of dataset variations, such as differences in image quality and patient demographics, on model performance, and suggest strategies for improving transferability across different clinical environments.

The benchmark framework proposed in this study provides valuable insights into the strengths and weaknesses of current DR detection methods. It encourages the development of more robust models that can be deployed in real-world settings where dataset differences and image quality can vary. The paper concludes with recommendations for future research, including the integration of multimodal data sources and the use of more sophisticated deep learning techniques to handle complex DR grading and segmentation tasks.

**CHAPTER 3**

**PROBLEM DEFINITION**

**3.1 EXISTING SYSTEM**

The existing systems for diabetic retinopathy (DR) detection primarily focus on classifying the grade of retinal images and extracting specific retinal features such as blood vessels, the optic disc, microaneurysms, hemorrhages, and exudates. These systems are typically used to aid ophthalmologists by automating parts of the diagnostic process. However, many of these systems are limited in functionality they often stop at feature extraction and classification without actually predicting the presence or severity of diabetic-related diseases. This limitation poses a challenge in providing comprehensive diagnostic support, especially when dealing with large-scale screenings where timely and accurate predictions are crucial. Recent developments in AI-based approaches have explored the use of Graph Neural Networks (GNNs) to enhance the structure-based understanding of retinal data. In these models, different features are represented as nodes, and their relationships as edges, enabling hierarchical learning. While GNNs offer improved interpretability and structural learning, they still come with limitations such as complex architecture, dependency on graph construction quality, and lack of direct disease prediction.

**3.1.1 DISADVANTAGES**

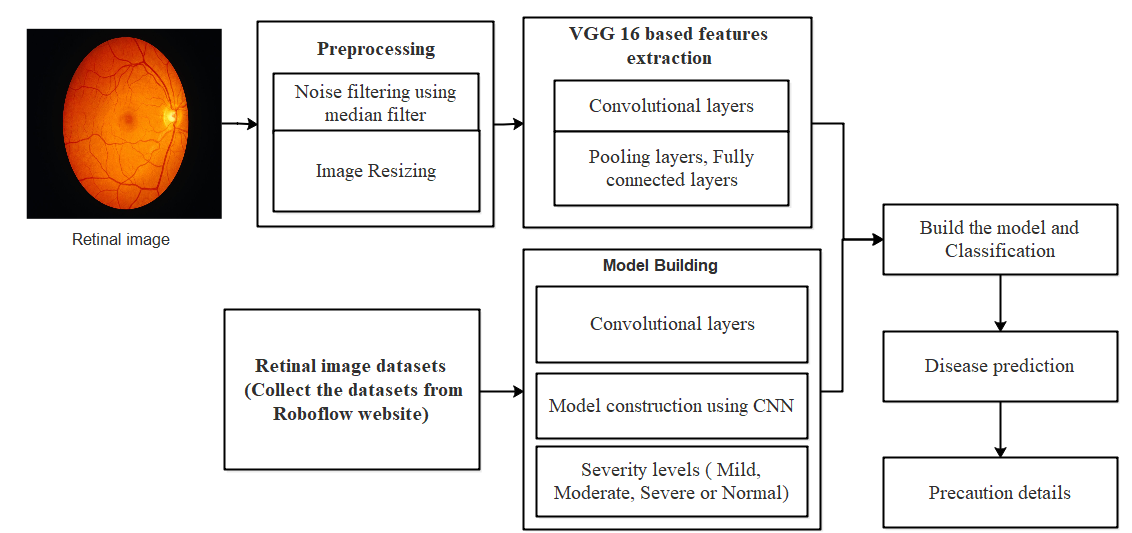
* Only classify the grade of uploaded images
* Extract the retinal features like vessels, optic disc, microaneurysms, hemorrhage, exudates (including hard exudates and soft exudates)
* There is no algorithm for predicting diseases
* Does not support large datasets

**CHAPTER 4**

**PROPOSED SYSTEM**

The proposed system introduces an advanced deep learning-based solution using the VGG16 Convolutional Neural Network (CNN) model for the prediction and classification of diabetic retinopathy and glaucoma from retinal images. This system begins with the acquisition of a comprehensive and diverse dataset containing retinal images labeled for diabetic and glaucoma conditions. The dataset is pre-processed through image enhancement techniques like noise reduction, contrast adjustment, and normalization to ensure clarity and consistency. Important regions such as optic disc, blood vessels, and dark lesions are isolated to emphasize pathological features critical for accurate diagnosis. Following preprocessing, the system uses VGG16 to automatically extract deep features from the processed retinal images. VGG16’s deep architecture allows it to capture intricate and hierarchical image patterns that signify various stages of diabetic retinopathy and glaucoma. The model is trained with these labeled images to distinguish between normal and abnormal retinal conditions effectively. Through hyperparameter tuning and optimization techniques, the CNN is fine-tuned to improve classification accuracy. Moreover, activation mapping techniques such as Grad-CAM or saliency maps are used to visually interpret the model’s focus areas, which enhances clinical trust in automated predictions. Finally, once trained and validated, the system is capable of real-time prediction, making it suitable for clinical deployment. The model not only detects the presence of diabetic retinopathy or glaucoma but also provides multi-level classification of the disease stages, aiding in early intervention and personalized treatment plans. By minimizing manual intervention and reducing diagnostic time, this automated solution improves scalability, especially in under-resourced or rural healthcare environments. It also addresses the black-box nature of CNNs by integrating interpretability techniques, ensuring that clinicians can make informed decisions based on model insights.

**4.1 ARCHITECTURE DIAGRAM**

****

In this architecture, the process is divided into two phases: training and testing. During the training phase, retinal image datasets, obtained from the KAGGLE website, are input into the system. These images are then processed using the Convolutional Neural Network (CNN) algorithm, which is trained to classify the images into five distinct disease categories. The testing phase begins by inputting new retinal images into the system, where noise filtering is applied to remove any unwanted artifacts, and the images are resized to fit the model’s requirements. The feature maps are then extracted using the trained CNN model, which are classified using a pre-trained model file. The system classifies the retinal images and predicts the presence of disease, achieving improved accuracy due to the refined preprocessing and feature extraction steps.

**4.1.2 ADVANTAGES**

* Implemented in real time
* Detect the diabetic diseases with multiple levels
* Dark objects are identified easily for Glaucoma detection
* Provide the diagnosis information

**4.2 ALGORITHM USED**

**VGG16 ALGORITHM**

VGG16 is a Convolutional Neural Network (CNN) architecture developed by the Visual Geometry Group (VGG) at Oxford. It is known for its simplicity and uniform architecture consisting of 13 convolutional layers and 3 fully connected layers. VGG16 uses small (3x3) convolutional filters throughout the network, which helps in extracting fine-grained features from images. In this project, VGG16 is used to analyze retinal images for detecting diabetic retinopathy and glaucoma. The deep layers of VGG16 enable it to learn complex visual representations, making it highly effective for medical image classification tasks. The model is pre-trained on ImageNet, allowing the system to leverage transfer learning by fine-tuning the existing weights for medical image inputs. This significantly reduces training time while improving accuracy. During training, the retinal images go through preprocessing steps such as resizing, normalization, and enhancement before being passed into the VGG16 model. The extracted features are then used to classify the images into healthy or diseased categories. This end-to-end architecture makes VGG16 a powerful backbone for developing an automated, accurate, and interpretable diagnostic tool for diabetic prediction.

**Advantages:**

* Provides high accuracy in image classification due to deep feature extraction.
* Uses transfer learning, reducing training time and data requirements.
* Handles complex patterns and fine details in medical images effectively.
* Architecture is simple and consistent, making implementation and tuning easier.

**DARK OBJECT DETECTION (IMAGE PREPROCESSING TECHNIQUE)**

Dark object detection is a preprocessing technique used to enhance the visibility of dark features in images, such as microaneurysms and optic disc shadows commonly seen in diabetic retinopathy and glaucoma. These features are critical indicators in retinal disease diagnosis. The method involves contrast enhancement and adaptive thresholding to highlight dark regions against the lighter background of the retina. By applying this technique, the system ensures that small, low-contrast features which are often missed in raw images become more prominent. In this project, dark object detection plays a crucial role in improving the input quality before it is fed into the CNN. Enhanced images improve the ability of the VGG16 model to detect pathological patterns with higher precision. This technique is especially beneficial in cases where retinal images are noisy, blurred, or vary in illumination, which is common in real-world datasets. When combined with segmentation and classification algorithms, dark object detection significantly contributes to the robustness of the entire diagnostic pipeline.

**Advantages:**

* Enhances visibility of small and dark retinal features like microaneurysms.
* Improves accuracy of the CNN by providing cleaner, high-contrast inputs.
* Handles image quality issues such as poor lighting or low contrast.
* Makes the model more robust to real-world medical image variations.

**4.3 MODULES**

* IMAGE ACQUISITION
* PREPROCESSING
* MODEL CONSTRUCTION
* CLASSIFICATION
* DISEASE DIAGNOSIS

**4.4 MODULES DESCRIPTION**

**4.4.1 IMAGE ACQUISITION:**

In this module is used to acquire a digital image. Retinal images of humans play an important role in the detection and diagnosis of cardiovascular diseases that including stroke, diabetes, arterio sclerosis, cardiovascular diseases and hypertension. Vascular diseases are often life critical for individuals, and present a challenging public health problem for society. The detection for retinal images is necessary and among them the detection of blood vessels is most important. The alterations about blood vessels such as length, width and branching pattern, can not only provide information on pathological changes but can also help to grade diseases severity or automatically diagnose the diseases. Upload the retinal images. The fundus of the eye is the interior surface of the eye, opposite the lens, and includes the retina, optic disc, macula and fovea, and posterior pole. The fundus can be examined by ophthalmoscope or fundus photography. The retina is a layered structure with several layers of neurons interconnected by synapses. In retina we can identify the vessels. Blood vessels show abnormalities at early stages also blood vessel alterations. Generalized arteriolar and venular narrowing which is related to the higher blood pressure levels, which is generally expressed by the Arteriolar to Venular diameter ratio. It constructed a dataset of images for the training and evaluation of our proposed method. This image dataset was acquired from publically available datasets such as DRIVE and STAR. Each image was captured using 24 bit per pixel (standard RGB) at 760 x 570 pixels. First, tested against normal images which are easier to distinguish. Second, some level of success with abnormal vessel appearances must be established to recommend clinical usage. As can be seen, a normal image consists of blood vessels, optic disc, fovea and the background, but the abnormal image also has multiple artifacts of distinct shapes and colors caused by different diseases.

Input retinal images

Retrieve retinal images

Check image quality

Storage image in database

**4.4.2 PREPROCESSING:**

To improve the image in ways that increases the chances for success of the other processes. The gray scale conversion operation is to identify black and white illumination. Noise in colored retinal image is normally due to noise pixels and pixels whose color is distorted so implement median filter can be used to enhance and sharpen the vascular pattern for preprocessing and blood vessel segmentation of retinal images performing well in preprocessing, enhancing and segmenting the retinal image and vascular pattern. Human perception is highly sensitive to edges and fine details of an image, and since they are composed primarily by high frequency components, the visual quality of an image can be enormously degraded if the high frequencies are attenuated or completed removed. In contrast, enhancing the high frequency components of an image leads to an improvement in the visual quality. Image sharpening refers to any enhancement technique that highlights edges and fine details in an image. Image sharpening is widely used in printing and photographic industries for increasing the local contrast and sharpening the images. In principle, image sharpening consists of adding to the original image a signal that is proportional to a high-pass filtered version of the original image. In this filter, the original image is first filtered by a high-pass filter that extracts the high-frequency components, and then a scaled version of the high-pass filter output is added to the original image, thus producing a sharpened image of the original. Note that the homogeneous regions of the signal, i.e., where the signal is constant, remain unchanged.

Input retinal images

Noise removal

Contrast enhancement

Image resizing

Normalization

**4.4.3 SEGMENTATION:**

Retinal image segmentation using CNN involves leveraging deep learning techniques to automatically identify and segment important regions in retinal images. The process begins with the collection of a large and diverse dataset of retinal images, ensuring that it includes variations in factors such as age, sex, and ethnicity. This ensures that the CNN algorithm can generalize across different populations. Preprocessing is then applied to enhance the quality of the images, using methods like noise reduction, contrast enhancement, and normalization to improve image clarity and reduce artifacts. Once the images are pre-processed, segmentation can take place to extract regions of interest like the optic nerve head and retinal blood vessels. A CNN model, such as VGG16, is then trained on the pre-processed images with labeled segmentation masks that highlight key regions. The CNN learns to automatically extract features and segment the images accordingly. The trained model is evaluated on a separate test dataset to ensure it generalizes well and performs accurately, and hyperparameter adjustments or transfer learning techniques can be applied for fine-tuning. By automating the segmentation of retinal images, healthcare providers can save time, reduce human errors, and improve diagnostic accuracy. This ultimately enhances patient outcomes by facilitating more reliable detection and analysis of retinal diseases, such as diabetic retinopathy, using a CNN model with VGG16.

Identify region of interest

Detect retinal features

Apply thresholding

Separate key features

Generate segmented output

**4.4.4 CLASSIFICATION:**

Diabetic and glaucoma classification using CNN focuses on employing deep learning techniques to automatically categorize retinal images into diabetic or non-diabetic categories, as well as glaucoma or non-glaucoma categories. The first step is data collection, where a large dataset of retinal images, labeled with corresponding diagnoses, is gathered. The dataset needs to include both diabetic and non-diabetic retinal images, as well as glaucoma and non-glaucoma retinal images. This dataset should be diverse to account for variations in factors such as age, sex, and ethnicity, ensuring the algorithm can generalize across different populations. Once the dataset is collected, it undergoes preprocessing to remove any noise and artifacts, and to enhance image quality. Techniques such as noise reduction, contrast enhancement, and normalization are applied. After preprocessing, the images are used to train a CNN model for classification, with labeled images indicating their respective diagnoses. The CNN model learns to extract important features from the images and classify them based on these features. Transfer learning can also be utilized to fine-tune the model and improve its performance. Once the model is trained, it is evaluated using a separate test dataset to assess its accuracy and generalizability. To improve the model's performance, optimization techniques such as hyperparameter adjustments and data augmentation can be applied. The classification process, powered by CNN and VGG16, helps automate the diagnosis of diabetic retinopathy and glaucoma, which reduces diagnostic errors and assists healthcare providers in making accurate and efficient decisions for better patient care.

Load trained VGG16

Feature extraction

Fully connected layers

Predict disease

Generate class label

**4.4.5 DISEASE DIAGNOSIS:**

The Disease Diagnosis Module for diabetic retinopathy and glaucoma involves a comprehensive deep learning approach for accurate detection and prediction. Diabetic retinopathy is a complication that occurs when high blood sugar levels damage the blood vessels in the retina, leading to vision loss or blindness. Glaucoma, on the other hand, refers to a group of eye disorders that cause damage to the optic nerve, often leading to vision loss. The primary goal of this module is to classify retinal images and predict the likelihood of these diseases using machine learning techniques. The process begins with data collection, where a large set of retinal images, labeled with their respective diagnoses (diabetic, non-diabetic, and glaucoma-related), is gathered. Preprocessing steps such as noise reduction, contrast enhancement, and normalization are applied to ensure the quality of the images before they are fed into the model. These preprocessed images are then used to train a deep learning model, which learns to extract relevant features for disease prediction. In the case of diabetic retinopathy, the model classifies the disease into levels such as No DR, Mild, Moderate, Severe, and Proliferative DR. Similarly, for glaucoma, the model predicts the likelihood of the disease based on the features extracted from the retinal images. Transfer learning can be employed to improve the model's performance, especially when fine-tuning for specific cases. Once the model is trained, it is evaluated using a separate test dataset to assess its generalizability and accuracy. Optimization techniques like hyperparameter tuning and data augmentation can be applied to further enhance the model’s effectiveness. By leveraging deep learning algorithms, this module automates disease diagnosis, providing reliable results that can assist healthcare professionals in making timely and accurate decisions for patient treatment and care.

Receive classification output

Match with severity level

Display diagnosis result

Store results

**CHAPTER 5**

**SYSTEM SPECIFICATION**

**5.1 HARDWARE REQUIREMENTS**

* Processor : Intel core processor 2.6.0 GHZ
* RAM : 1GB
* Hard disk : 160 GB
* Compact Disk : 650 Mb
* Keyboard : Standard keyboard
* Monitor : 15 inch color monitor

**5.2 SOFTWARE REQUIREMENTS**

* Operating system : Windows OS
* Front End : PYTHON
* IDE : PYCHARM
* Libraries : Tensorflow, KERAS

D **FRONT END: PYTHON**

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages due to its simplicity, readability, and versatility. One of the key features of Python is its easy-to-learn syntax, which makes it accessible to both novice and experienced programmers. It has a large standard library that provides a wide range of modules for tasks such as file I/O, networking, regular expressions, and more. Python also has a large and active community of developers who contribute to open source libraries and packages that extend its capabilities. Python is an interpreted language, which means that it is executed line-by-line by an interpreter rather than compiled into machine code like C or C++. This allows for rapid development and testing, as well as easier debugging and maintenance of code. Python is used for a variety of applications, including web development frameworks such as Django and Flask, scientific computing libraries such as NumPy and Pandas, and machine learning libraries such as TensorFlow and PyTorch. It is also commonly used for scripting and automation tasks due to its ease of use and readability. Overall, Python is a powerful and versatile programming language that is widely used in a variety of domains due to its simplicity, ease of use, and active community.



FIG: PYTHON IMPLEMENTATION

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture."Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity.When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar. A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

Python also has a large and active community of developers who contribute to a wide range of open-source libraries and tools, making it easy to find and use pre-built code to solve complex problems.

Python has a wide range of applications, including:

Data Science: Python is one of the most popular languages for data science, thanks to libraries like NumPy, Pandas, and Matplotlib that make it easy to manipulate and visualize data.

Machine Learning: Python is also widely used in machine learning and artificial intelligence, with libraries like TensorFlow, Keras, and Scikit-learn that provide powerful tools for building and training machine learning models.

Web Development: Python is commonly used in web development, with frameworks like Django and Flask that make it easy to build web applications and APIs.

Scientific Computing: Python is used extensively in scientific computing, with libraries like SciPy and SymPy that provide powerful tools for numerical analysis and symbolic mathematics.

In addition to its versatility and ease of use, Python is also known for its portability and compatibility. Python code can be run on a wide range of platforms, including Windows, macOS, and Linux, and it can be integrated with other languages like C and Java.

Overall, Python is a powerful and versatile programming language that is well-suited for a wide range of applications, from data science and machine learning to web development and scientific computing. Its simplicity, readability, and large community of developers make it an ideal choice for beginners and experts alike.

One of the strengths of Python is its rich ecosystem of third-party libraries and tools. These libraries provide a wide range of functionality, from scientific computing and data analysis to web development and machine learning. Some popular Python libraries and frameworks include:

NumPy: a library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.

Pandas: a library for data manipulation and analysis in Python, providing support for reading and writing data in a variety of formats, as well as powerful tools for manipulating and analyzing data.

Matplotlib: a plotting library for Python that provides a variety of visualization tools, including line plots, scatter plots, bar plots, and more.

TensorFlow: an open-source machine learning library for Python that provides a variety of tools and algorithms for building and training machine learning models.

Django: a popular web framework for Python that provides a full-stack framework for building web applications, with support for everything from URL routing to user authentication and database integration.

Python's popularity has also led to a large and active community of developers who contribute to open-source projects and share code and resources online. This community provides a wealth of resources for learning Python, including tutorials, online courses, and forums for asking and answering questions.

Overall, Python is a versatile and powerful programming language that is well-suited for a wide range of applications. Its simplicity, flexibility, and wide range of libra

**TENSORFLOW LIBARIES IN PYTHON**

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It is one of the most popular libraries for building and training machine learning models, especially deep neural networks. TensorFlow allows developers to build complex models with ease, including image and speech recognition, natural language processing, and more. One of the key features of TensorFlow is its ability to handle large-scale datasets and complex computations, making it suitable for training deep neural networks. It allows for parallelization of computations across multiple CPUs or GPUs, allowing for faster training times. TensorFlow also provides a high-level API called Keras that simplifies the process of building and training models. TensorFlow offers a wide range of tools and libraries that make it easy to integrate with other Python libraries and frameworks. It has built-in support for data preprocessing and visualization, making it easy to prepare data for training and analyze model performance. One of the major advantages of TensorFlow is its ability to deploy models to a variety of platforms, including mobile devices and the web.

Graph-based computation: TensorFlow uses a graph-based computation model, which allows for efficient execution of computations across multiple devices and CPUs/GPUs.

Automatic differentiation: TensorFlow provides automatic differentiation, which allows for efficient computation of gradients for use in backpropagation algorithms.

High-level APIs: TensorFlow provides high-level APIs, such as Keras, that allow developers to quickly build and train complex models with minimal code.

Preprocessing and data augmentation: TensorFlow provides a range of tools for preprocessing and data augmentation, including image and text preprocessing, data normalization, and more.

Distributed training: TensorFlow supports distributed training across multiple devices, CPUs, and GPUs, allowing for faster training times and more efficient use of resources.

Model deployment: TensorFlow allows for easy deployment of models to a variety of platforms, including mobile devices and the web.

Visualization tools: TensorFlow provides a range of visualization tools for analyzing model performance, including TensorBoard, which allows for real-time visualization of model training and performance.

**PYCHARM**

PyCharm is an integrated development environment (IDE) for Python programming language, developed by JetBrains. PyCharm provides features such as code completion, debugging, code analysis, refactoring, version control integration, and more to help developers write, test, and debug their Python code efficiently. PyCharm is available in two editions: Community Edition (CE) and Professional Edition (PE). The Community Edition is a free, open-source version of the IDE that provides basic functionality for Python development. The Professional Edition is a paid version of the IDE that provides advanced features such as remote development, web development, scientific tools, database tools, and more. PyCharm is available for Windows, macOS, and Linux operating systems. It supports Python versions 2.7, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, and 3.10.

**Features:**

* Intelligent code completion
* Syntax highlighting
* Code inspection
* Code navigation and search
* Debugging
* Testing
* Version control integration
* Web development support
* Scientific tools support
* Database tools support

**Integration with other JetBrains tools**

PyCharm's code completion feature can help speed up development by automatically suggesting code based on context and previously written code. It also includes a debugger that allows developers to step through code, set breakpoints, and inspect variables. PyCharm has integration with version control systems like Git, Mercurial, and Subversion. It also supports virtual environments, which allow developers to manage different Python installations and packages in isolated environments. The IDE also has features specifically geared towards web development, such as support for popular web frameworks like Django, Flask, and Pyramid. It includes tools for debugging, testing, and profiling web applications. PyCharm also provides scientific tools for data analysis, visualization, and scientific computing, such as support for NumPy, SciPy, and matplotlib. It also includes tools for working with databases, such as PostgreSQL, MySQL, and Oracle. Overall, PyCharm is a powerful and feature-rich IDE that can greatly increase productivity for Python developers.

**Customization:**

PyCharm allows developers to customize the IDE to their liking. Users can change the color scheme, fonts, and other settings to make the IDE more comfortable to use. PyCharm also supports plugins, which allow developers to extend the IDE with additional features.

**Collaboration:**

PyCharm makes it easy for developers to collaborate on projects. It supports integration with popular collaboration tools such as GitHub, Bitbucket, and GitLab. It also includes features for code reviews, task management, and team communication.

**Education:**

PyCharm provides a learning environment for Python programming language. PyCharm Edu is a free, open-source edition of PyCharm that includes interactive courses and tutorials for learning Python. It provides an easy-to-use interface for beginners and includes features such as code highlighting, autocompletion, and error highlighting.

**Support:**

PyCharm has an active community of users who provide support through forums and social media. JetBrains also provides comprehensive documentation, tutorials, and training courses for PyCharm. For users who need more personalized support, JetBrains offers a paid support plan that includes email and phone support.

**Pricing:**

PyCharm Community Edition is free and open-source. PyCharm Professional Edition requires a paid license, but offers a 30-day free trial. JetBrains also offers a subscription-based pricing model that includes access to all JetBrains IDEs and tools.

**Integrations:**

PyCharm integrates with a wide range of tools and technologies commonly used in Python development. It supports popular Python web frameworks like Flask, Django, Pyramid, and web2py. It also integrates with tools for scientific computing like NumPy, SciPy, and pandas. PyCharm also supports popular front-end technologies such as HTML, CSS, and JavaScript.

**Performance:**

PyCharm is known for its fast and reliable performance. It uses a combination of static analysis, incremental compilation, and intelligent caching to provide fast code completion and navigation. PyCharm also has a memory profiler that helps identify and optimize memory usage in Python applications.

**Ease of Use:**

PyCharm provides an intuitive and easy-to-use interface for developers. It has a well-organized menu structure, clear icons, and easy-to-navigate tabs. PyCharm also provides a variety of keyboard shortcuts and customizable keymaps that allow users to work efficiently without constantly switching between the mouse and keyboard.

**Community:**

PyCharm has a large and active community of developers who contribute to the development of the IDE. The PyCharm Community Edition is open-source, which means that anyone can contribute to its development. The PyCharm user community is also active in providing support, tips, and tutorials through forums, blogs, and social media.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 CONCLUSION**

In conclusion, the proposed AI-driven diabetic diagnosis system leveraging the VGG16 convolutional neural network architecture offers a robust solution for the automated detection of diabetic retinopathy and glaucoma. By processing retinal images through a series of preprocessing, feature extraction, and classification steps, the system significantly enhances diagnostic accuracy while reducing the reliance on manual examination. The use of advanced image processing techniques, such as dark object detection and noise filtering, ensures that even subtle abnormalities in the retina are accurately detected, contributing to earlier diagnosis and intervention. This automated system not only improves the efficiency of diagnosis but also facilitates scalable screening, particularly in areas with limited access to ophthalmologists. By leveraging the power of deep learning and transfer learning, the system can handle large datasets and achieve high performance with reduced training time. As a result, it holds great potential in improving the overall healthcare system by providing quicker, more reliable diagnostic support, ultimately leading to better patient outcomes in diabetic care.

**6.2 FUTURE WORK**

* **Incorporating Multi-Disease Detection:** The system can be extended to detect multiple diabetic-related diseases beyond diabetic retinopathy and glaucoma, such as macular edema and diabetic cataract, for a more comprehensive diagnosis.
* **Real-Time Diagnosis:** Implementing the system in real-time settings to provide instant diagnostic results for retinal images captured by portable devices, allowing for immediate feedback in clinical environments.
* **Integration with Electronic Health Records (EHR):** Integrating the system with patient health records to provide a more holistic approach to disease monitoring and management, allowing healthcare professionals to track patients’ progress over time.
* **Improved Preprocessing Techniques:** Enhancing the image preprocessing phase by integrating more advanced techniques, such as deep learning-based denoising or advanced contrast enhancement, to further improve the quality of the input images.
* **Adaptation for Different Populations:** Training the model on diverse datasets from various ethnic groups and age ranges to ensure its accuracy and applicability to a broader population.
* **Incorporation of 3D Retinal Imaging:** Extending the system to handle 3D retinal imaging technologies, such as optical coherence tomography (OCT), for more detailed analysis of the retina and better disease detection.
* **Explainability and Interpretability:** Incorporating explainable AI techniques, such as Grad-CAM or saliency mapping, to make the model’s predictions more transparent, helping healthcare professionals understand the rationale behind the diagnosis.
* **Automated Follow-up System:** Developing an automated system for scheduling follow-up appointments or recommending treatment based on the severity of the detected disease, improving patient care and treatment adherence.

**APPENDICES**

**A1. SAMPLE SOURCE CODE**

from tkinter import \*

import os

from tkinter import filedialog

import cv2

from tkinter import messagebox

def file\_sucess():

global file\_success\_screen

file\_success\_screen = Toplevel(training\_screen)

file\_success\_screen.title("File Upload Success")

file\_success\_screen.geometry("150x100")

Label(file\_success\_screen, text="File Upload Success").pack()

Button(file\_success\_screen, text='''ok''', font=(

'Palatino Linotype', 15), height="2", width="30").pack()

global ttype

def training():

global training\_screen

global clicked

training\_screen = Toplevel(main\_screen)

training\_screen.title("Training")

# login\_screen.geometry("400x300")

training\_screen.geometry("600x450+650+150")

training\_screen.minsize(120, 1)

training\_screen.maxsize(1604, 881)

training\_screen.resizable(1, 1)

training\_screen.configure()

# login\_screen.title("New Toplevel")

Label(training\_screen, text='''Upload Image ''',

foreground="#000000", width="300", height="2", font=("Palatino Linotype", 16)).pack()

Label(training\_screen, text="").pack()

options = [

'0NoDR', '1Mild', '2Moderate', '3Severe', '4ProliferativeDR'

]

# datatype of menu text

clicked = StringVar()

# initial menu text

clicked.set("Normal")

# Create Dropdown menu

drop = OptionMenu(training\_screen, clicked, \*options)

drop.config(width="30")

drop.pack()

ttype = clicked.get()

Button(training\_screen, text='''Upload Image''', font=(

'Palatino Linotype', 15), height="2", width="30", command=imgtraining).pack()

def imgtraining():

name1 = clicked.get()

print(name1)

import\_file\_path = filedialog.askopenfilename()

import os

s = import\_file\_path

os.path.split(s)

os.path.split(s)[1]

splname = os.path.split(s)[1]

image = cv2.imread(import\_file\_path)

# filename = 'Test.jpg'

filename = 'Data/' + name1 + '/' + splname

cv2.imwrite(filename, image)

print("After saving image:")

image = cv2.resize(image, (780, 540))

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cv2.imshow('Original image', image)

cv2.imshow('Gray image', gray)

# import\_file\_path = filedialog.askopenfilename()

print(import\_file\_path)

fnm = os.path.basename(import\_file\_path)

print(os.path.basename(import\_file\_path))

from PIL import Image, ImageOps

im = Image.open(import\_file\_path)

im\_invert = ImageOps.invert(im)

im\_invert.save('lena\_invert.jpg', quality=95)

im = Image.open(import\_file\_path).convert('RGB')

im\_invert = ImageOps.invert(im)

im\_invert.save('tt.png')

image2 = cv2.imread('tt.png')

image2 = cv2.resize(image2, (540, 540))

cv2.imshow("Invert", image2)

""""-----------------------------------------------"""

img = image

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow('Original image', img)

dst = cv2.medianBlur(img, 7)

cv2.imshow("Nosie Removal", dst)

def fulltraining():

import Model as mm

def testing():

global testing\_screen

testing\_screen = Toplevel(main\_screen)

testing\_screen.title("Testing")

# login\_screen.geometry("400x300")

testing\_screen.geometry("600x450+650+150")

testing\_screen.minsize(120, 1)

testing\_screen.maxsize(1604, 881)

testing\_screen.resizable(1, 1)

testing\_screen.configure()

# login\_screen.title("New Toplevel")

Label(testing\_screen, text='''Upload Image''', width="300", height="2", font=("Palatino Linotype", 16)).pack()

Label(testing\_screen, text="").pack()

Label(testing\_screen, text="").pack()

Label(testing\_screen, text="").pack()

Button(testing\_screen, text='''Upload Image''', font=(

'Palatino Linotype', 15), height="2", width="30", command=imgtest).pack()

global affect

def imgtest():

import\_file\_path = filedialog.askopenfilename()

image = cv2.imread(import\_file\_path)

print(import\_file\_path)

filename = 'Output/Out/Test.jpg'

cv2.imwrite(filename, image)

print("After saving image:")

# result()

# import\_file\_path = filedialog.askopenfilename()

print(import\_file\_path)

fnm = os.path.basename(import\_file\_path)

print(os.path.basename(import\_file\_path))

# file\_sucess()

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nImage : " + fnm + "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

img = cv2.imread(import\_file\_path)

if img is None:

print('no data')

img1 = cv2.imread(import\_file\_path)

print(img.shape)

img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))

original = img.copy()

neworiginal = img.copy()

img1 = cv2.resize(img1, (540, 540))

cv2.imshow('original', img1)

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

img1S = cv2.resize(img1, (540, 540))

cv2.imshow('Original image', img1S)

grayS = cv2.resize(gray, (540, 540))

cv2.imshow('Gray image', grayS)

dst = cv2.fastNlMeansDenoisingColored(img1, None, 10, 10, 7, 21)

dst = cv2.resize(dst, (540, 540))

cv2.imshow("Nosie Removal", dst)

result()

def result():

import warnings

warnings.filterwarnings('ignore')

import tensorflow as tf

classifierLoad = tf.keras.models.load\_model('Model/diabetic.h5')

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('./Output/Out/Test.jpg', target\_size=(200, 200))

# test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis=0)

result = classifierLoad.predict(test\_image)

print(result)

out = ''

if result[0][0] == 1:

out = "NoDR"

messagebox.showinfo("Result", "Classification Result : " + str(out))

elif result[0][1] == 1:

out = "Mild"

messagebox.showinfo("Result", "Classification Result : " + str(out))

messagebox.showinfo("prescription", 'Short-acting (regular) insulin')

elif result[0][2] == 1:

out = "Moderate"

messagebox.showinfo("Result", "Classification Result : " + str(out))

messagebox.showinfo("prescription", 'Possibly, diabetes medication or insulin therapy')

elif result[0][3] == 1:

out = "Severe"

messagebox.showinfo("Result", "Classification Result : " + str(out))

messagebox.showinfo("prescription", 'Medicines called anti-VEGF drugs can slow down or reverse diabetic retinopathy')

elif result[0][4] == 1:

out = "ProliferativeDR"

messagebox.showinfo("Result", "Classification Result : " + str(out))

messagebox.showinfo("prescription", 'Severe proliferative retinopathy may be treated with a more aggressive laser therapy called scatter (pan-retinal) photocoagulation.')

def main\_account\_screen():

global main\_screen

main\_screen = Tk()

width = 600

height = 500

screen\_width = main\_screen.winfo\_screenwidth()

screen\_height = main\_screen.winfo\_screenheight()

x = (screen\_width / 2) - (width / 2)

y = (screen\_height / 2) - (height / 2)

main\_screen.geometry("%dx%d+%d+%d" % (width, height, x, y))

main\_screen.resizable(0, 0)

# main\_screen.geometry("300x250")

main\_screen.configure()

main\_screen.title("Diabetes Prediction ")

Label(text="Diabetes Prediction ", width="300", height="5", font=("Palatino Linotype", 16)).pack()

Button(text="UploadImage", font=(

'Palatino Linotype', 15), height="2", width="20", command=training, highlightcolor="black").pack(side=TOP)

Label(text="").pack()

Button(text="Training", font=(

'Palatino Linotype', 15), height="2", width="20", command=fulltraining, highlightcolor="black").pack(side=TOP)

Label(text="").pack()

Button(text="Testing", font=(

'Palatino Linotype', 15), height="2", width="20", command=testing).pack(side=TOP)

Label(text="").pack()

main\_screen.mainloop()

main\_account\_screen()

import matplotlib.pyplot as plt

import warnings

import seaborn as sns

import numpy

warnings.filterwarnings('ignore')

batch\_size = 32

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale=1/255)

train\_generator = train\_datagen.flow\_from\_directory('Data/Train',target\_size=(200, 200), batch\_size=batch\_size,

classes = ['0NoDR','1Mild','2Moderate','3Severe','4ProliferativeDR'],class\_mode='categorical')

test\_datagen = ImageDataGenerator(rescale=1/255)

test\_generator = test\_datagen.flow\_from\_directory('Data/Test', target\_size=(200, 200), batch\_size=batch\_size,

classes = ['0NoDR','1Mild','2Moderate','3Severe','4ProliferativeDR'],

class\_mode='categorical',shuffle=False)

import tensorflow as tf

model = tf.keras.models.Sequential([

# The first convolution

tf.keras.layers.Conv2D(16, (3,3), activation='relu', input\_shape=(200, 200, 3)),

tf.keras.layers.MaxPooling2D(2, 2),

# The second convolution

tf.keras.layers.Conv2D(32, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# The third convolution

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# The fourth convolution

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# The fifth convolution

tf.keras.layers.Conv2D(128, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# Flatten the results to feed into a dense layer

tf.keras.layers.Flatten(),

# 128 neuron in the fully-connected layer

tf.keras.layers.Dense(128, activation='relu'),

# 5 output neurons for 3 classes with the softmax activation

tf.keras.layers.Dense(5, activation='softmax')

])

model.summary()

from tensorflow.keras.optimizers import RMSprop

early = tf.keras.callbacks.EarlyStopping(monitor='val\_loss',patience=5)

model.compile(loss='categorical\_crossentropy', optimizer=RMSprop(lr=0.001),metrics=['accuracy'])

total\_sample=train\_generator.n

n\_epochs = 100

history = model.fit\_generator(train\_generator,steps\_per\_epoch=int(total\_sample/batch\_size),epochs=n\_epochs)

model.save('diabetic.h5')

acc = history.history['accuracy']

loss = history.history['loss']

epochs = range(1, len(acc) + 1)

# Train and validation accuracy

plt.plot(epochs, acc, 'b', label=' accurarcy')

plt.title('accurarcy')

plt.legend()

plt.figure()

# Train and validation loss

plt.plot(epochs, loss, 'b', label=' loss')

plt.title(' loss')

plt.legend()

plt.show()

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

test\_steps\_per\_epoch = numpy.math.ceil(test\_generator.samples / test\_generator.batch\_size)

predictions = model.predict\_generator(test\_generator, steps=test\_steps\_per\_epoch)

# Get most likely class

predicted\_classes = numpy.argmax(predictions, axis=1)

true\_classes = test\_generator.classes

class\_labels = list(test\_generator.class\_indices.keys())

print('Classification Report')

report = classification\_report(true\_classes, predicted\_classes, target\_names=class\_labels)

print(report)

print('confusion matrix')

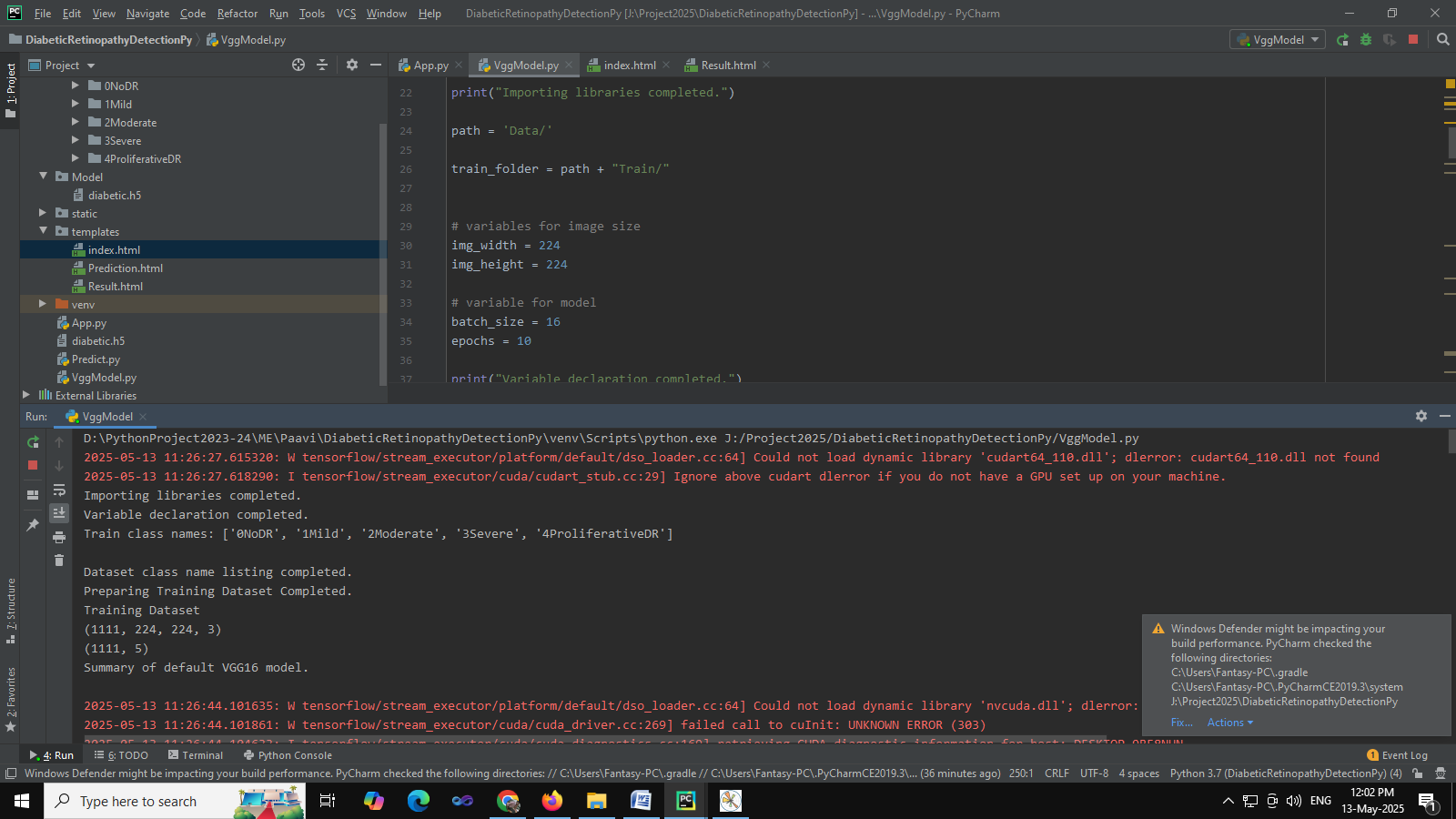
confusion\_matrix= confusion\_matrix(true\_classes, predicted\_classes)

print(confusion\_matrix)

sns.heatmap(confusion\_matrix, annot = True)

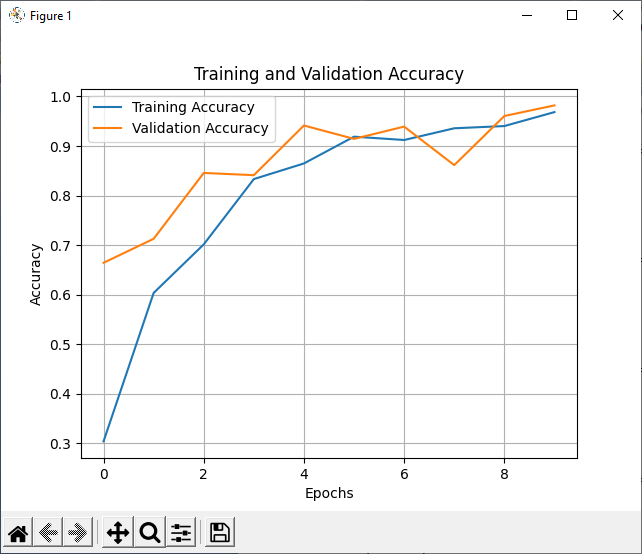
plt.show()

**A2. SAMPLE SCREENSHOTS**



56/56 [==============================] - 203s 4s/step - loss: 0.1477 - accuracy: 0.9685 - val\_loss: 0.1170 - val\_accuracy: 0.9820

Fitting the model completed.





precision recall f1-score support

0 0.99 1.00 1.00 239

1 0.99 0.98 0.99 231

2 0.99 0.97 0.98 232

3 0.92 1.00 0.96 108

4 1.00 0.92 0.96 78

accuracy 0.98 888

macro avg 0.98 0.98 0.98 888

weighted avg 0.98 0.98 0.98 888

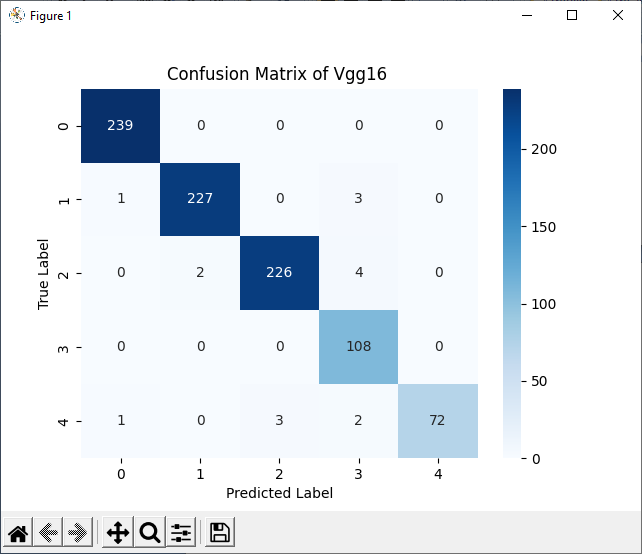
[[239 0 0 0 0]

[ 1 227 0 3 0]

[ 0 2 226 4 0]

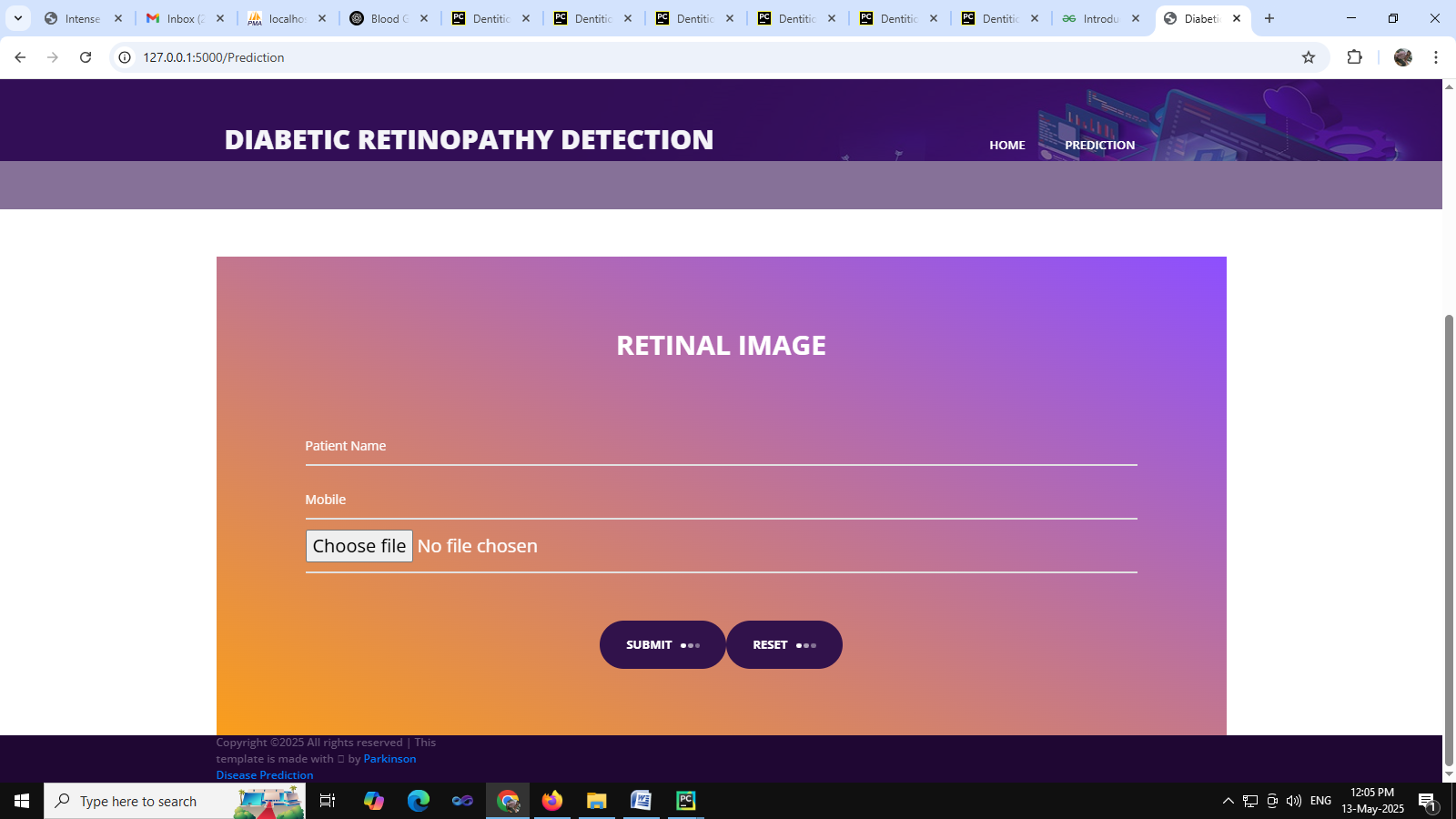
[ 0 0 0 108 0]

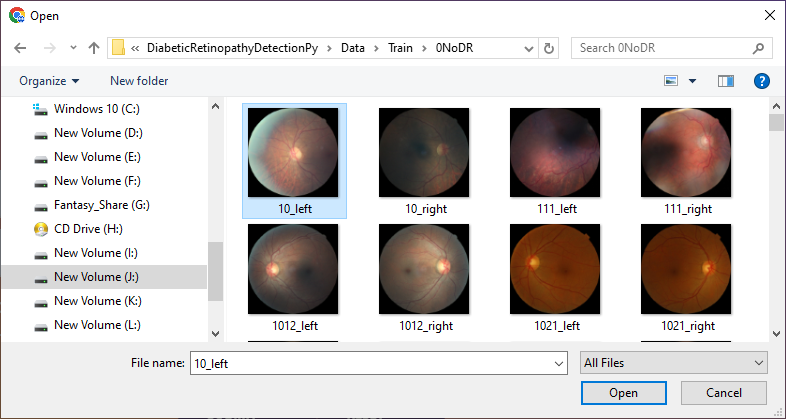
[ 1 0 3 2 72]]

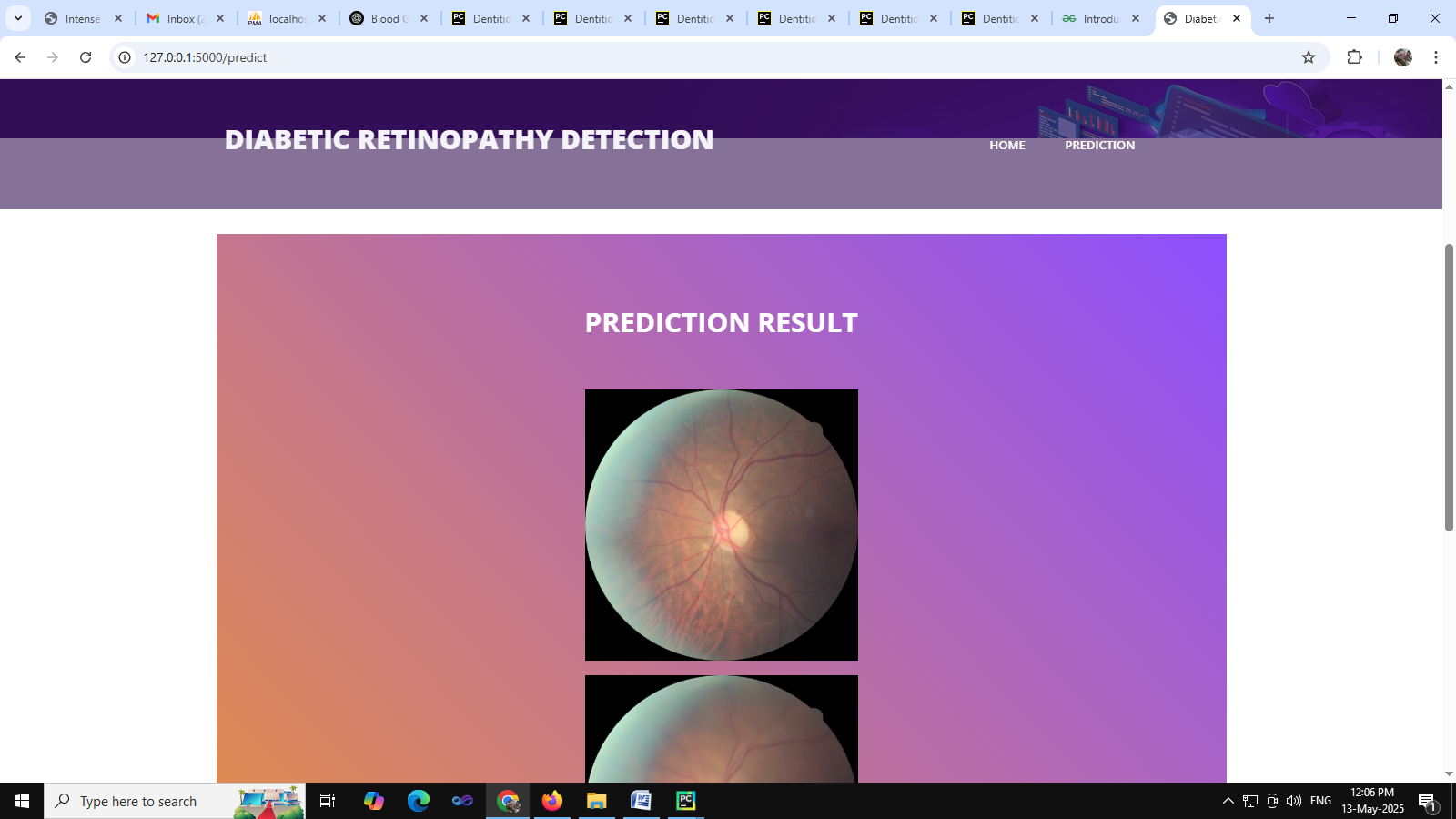


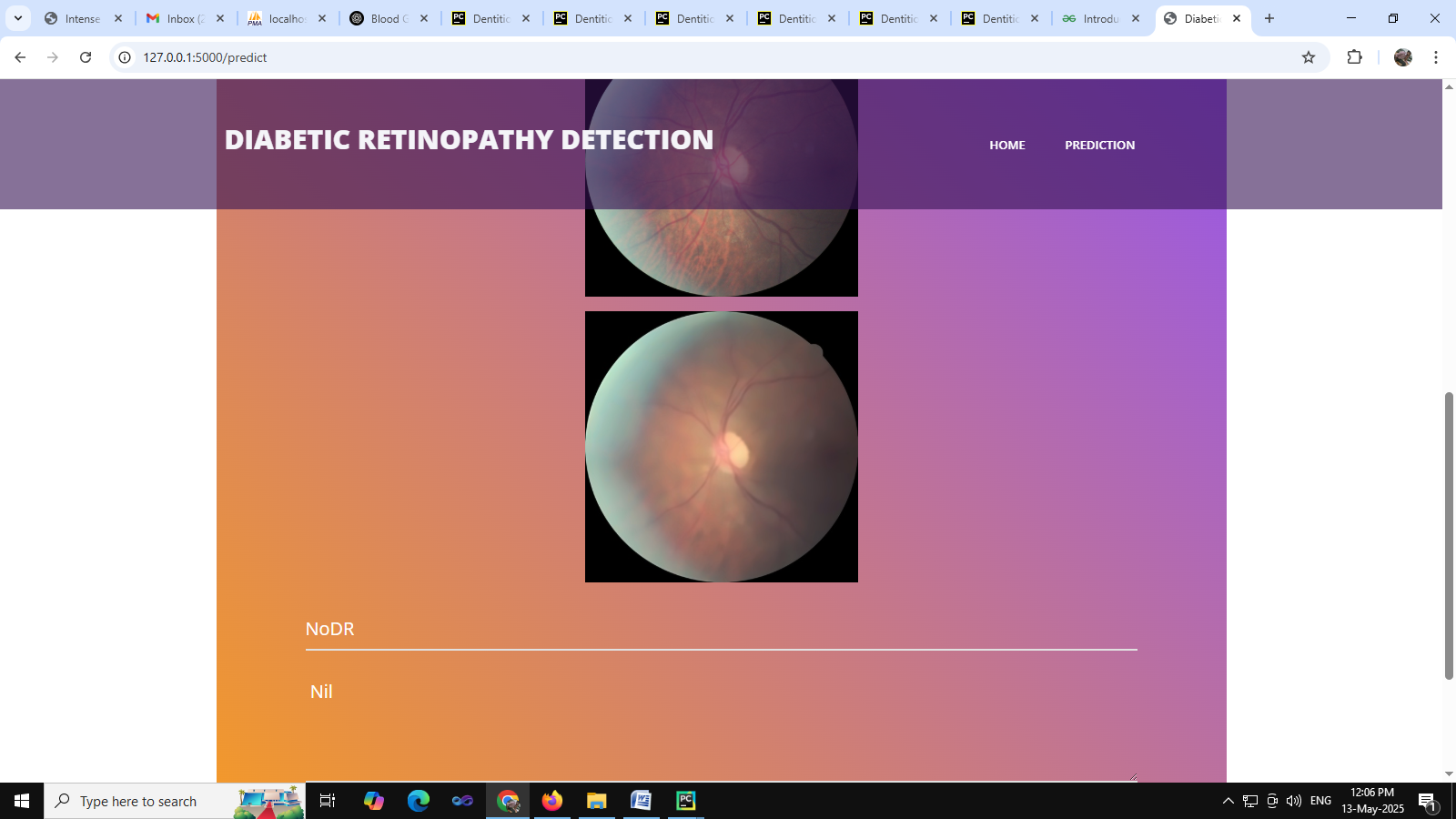


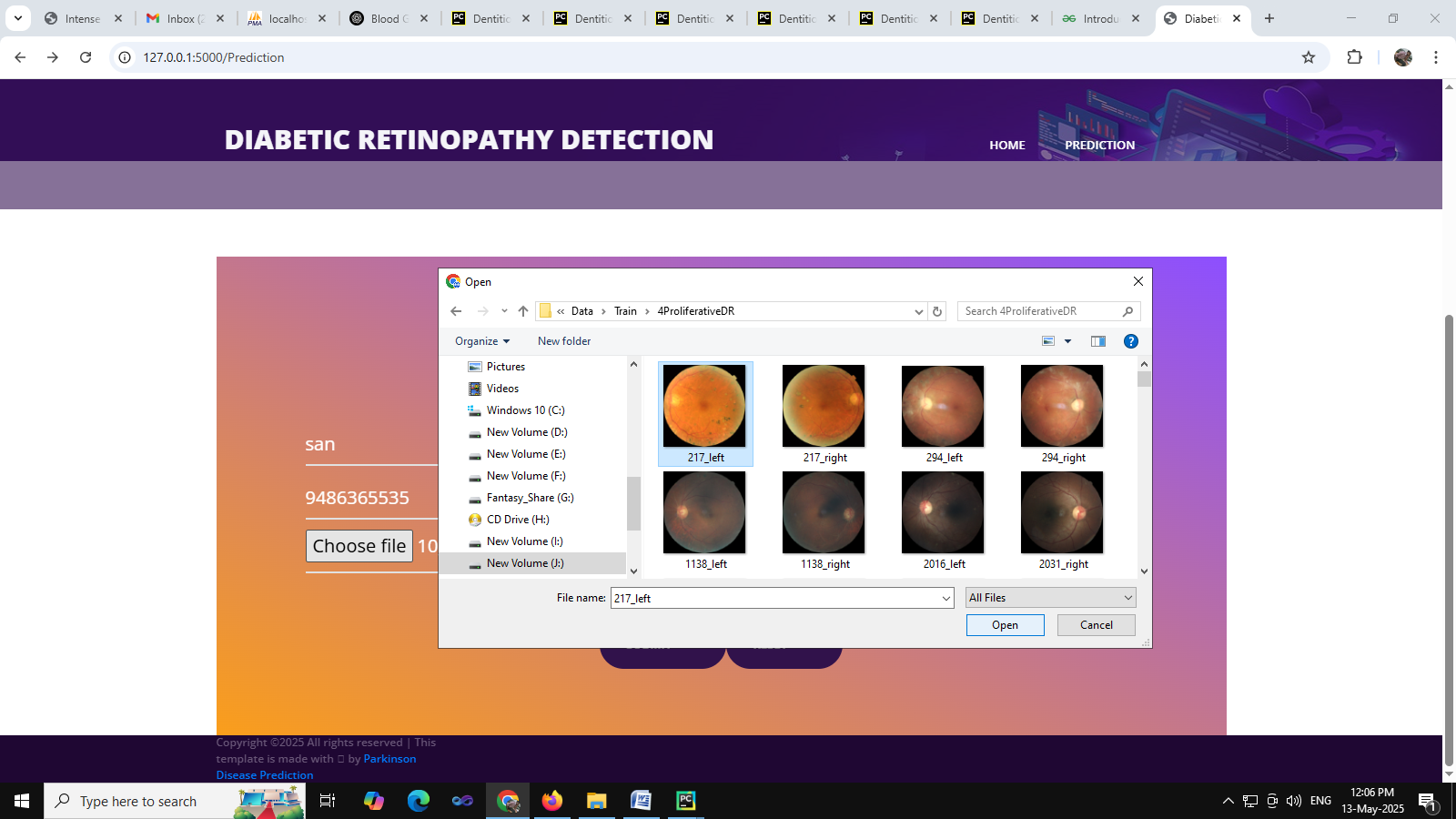


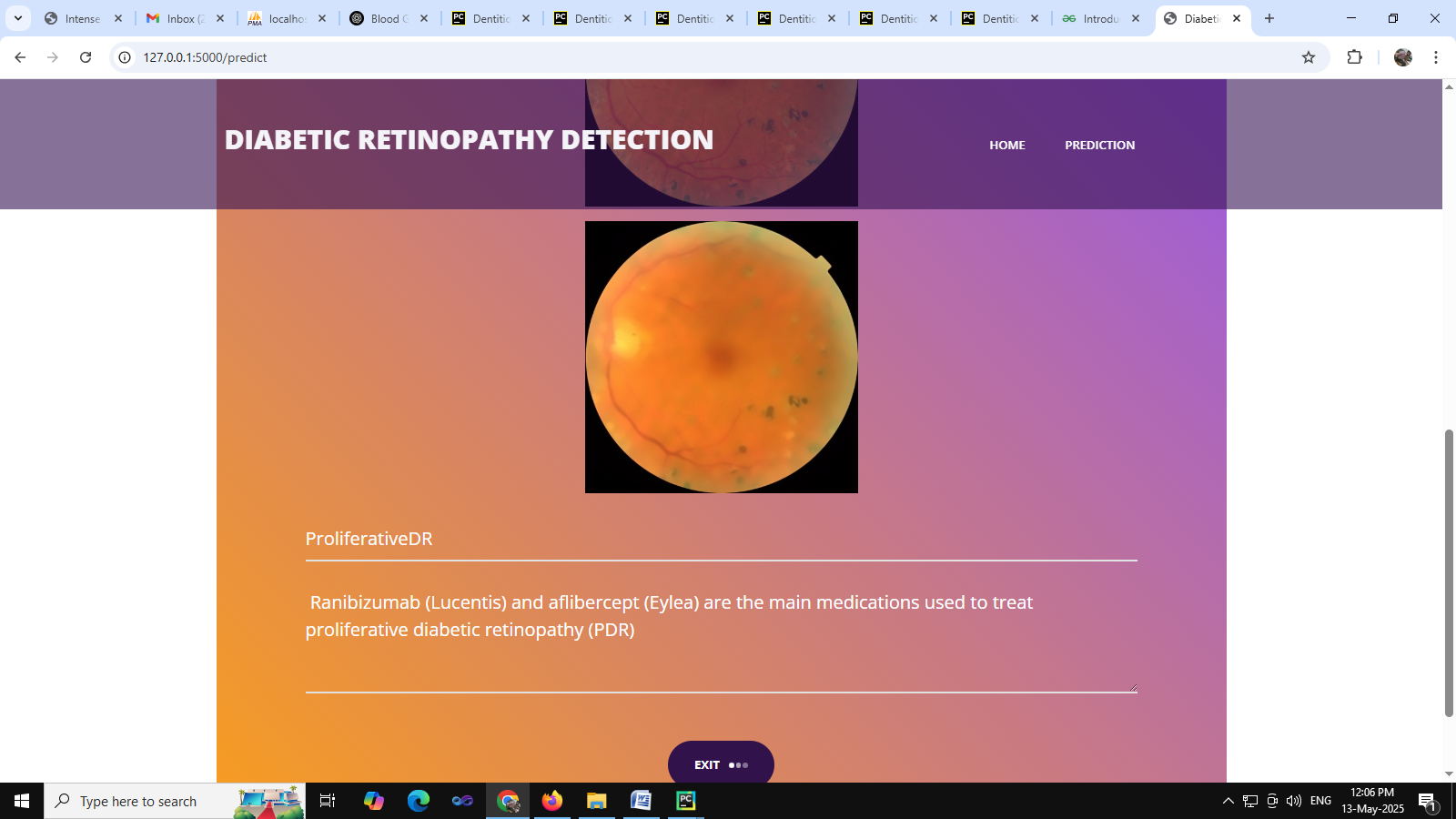












**REFERENCES**

[1] Mateen, Muhammad, et al. "Automatic detection of diabetic retinopathy: a review on datasets, methods and evaluation metrics." IEEE Access 8 (2020): 48784-48811.

[2] Shankar, K., et al. "Hyperparameter tuning deep learning for diabetic retinopathy fundus image classification." IEEE Access 8 (2020): 118164-118173.

[3] He, Along, et al. "CABNet: category attention block for imbalanced diabetic retinopathy grading." IEEE Transactions on Medical Imaging 40.1 (2020): 143-153.

[4] Kaushik, Harshit, et al. "Diabetic retinopathy diagnosis from fundus images using stacked generalization of deep models." IEEE Access 9 (2021): 108276-108292.

[5] Zhou, Yi, et al. "A benchmark for studying diabetic retinopathy: segmentation, grading, and transferability." IEEE Transactions on Medical Imaging 40.3 (2020): 818-828.

[6] Gayathri, S., et al. "Automated binary and multiclass classification of diabetic retinopathy using haralick and multiresolution features." IEEE Access 8 (2020): 57497-57504.

[7] Al-Antary, Mohammad T., and Yasmine Arafa. "Multi-scale attention network for diabetic retinopathy classification." IEEE Access 9 (2021): 54190-54200.

[8] Zhou, Yi, et al. "DR-GAN: conditional generative adversarial network for fine-grained lesion synthesis on diabetic retinopathy images." IEEE Journal of Biomedical and Health Informatics (2020).

[9] Abdelmaksoud, Eman, et al. "Automatic diabetic retinopathy grading system based on detecting multiple retinal lesions." IEEE Access 9 (2021): 15939-15960.

[10] Araújo, Teresa, et al. "Data augmentation for improving proliferative diabetic retinopathy detection in eye fundus images." IEEE Access 8 (2020): 182462-182474.