A PROJECT REPORT ON MULTIPLE EYE DISEASE CLASSIFICATION USING DEEP LEARNING

Submitted in Partial fulfillment of Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
CSE – (AI & ML)

Submitted By

K. KOSHAL	**(20MH1A4236)**
D. SAI MANOHAR	**(20MH1A4207)**
K. LEELA PRASAD	**(20MH1A4228)**
G. ARUN KUMAR	**(20MH1A4215)**

Under the Esteemed Guidance of
Y. Suresh Kumar, M.Tech.
Assistant Professor

DEPARTMENT OF CSE(AI&ML)

ADITYA COLLEGE OF ENGINEERING

Approved by AICTE, permanently affiliated to JNTUK & Accredited by NAAC Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956 Aditya Nagar, ADB Road – Surampalem 533437, E.G. Dist., A.P.

2020-2024

ADITYA COLLEGE OF ENGINEERING

Approved by AICTE, permanently affiliated to JNTUK & Accredited by NAAC Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956 Aditya Nagar, ADB Road –Surampalem 533437, E.G. Dist., A.P.
2020-2024

DEPARTMENT OF CSE - (AI & ML)

CERTIFICATE

This is to certify that the project work entitled, "**MULTIPLE EYE DISEASE CLASSIFICATION USING DEEP LEARNING**", is a work carried out by **K. KOSHAL (20MH1A4236), D. SAI MANOHAR (20MH1A4207), K. LEELA PRASAD (20MH1A4228), G. ARUN KUMAR (20MH1A4215).** in partial fulfillment of the requirement for the **award of the degree of Bachelor of Technology** in **CSE** - (**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**).

Project Guide **Head of the Department**

External Examiner

ACKNOWLEDGEMENT

We owe a great many thanks to a great many people who helped and supported and suggested us in every step.

We are glad to have the support of our principal **Dr. A RAMESH, Principal** who inspired us with his words filled with dedication and discipline towards work.

We express our gratitude towards **Dr. B. Kiran Kumar, M.Tech, PhD., Professor & HOD of CSE - (AI & ML)** for extending his support through training classes which had been the major source to carry our project.

We are very much thankful to **Mr. Y. Suresh Kumar, M.Tech, Assistant Professor**, Guide of our project for guiding and correcting various documents of ours with attention and care. She has taken pain to go through the project and make necessary corrections as and when needed.

Finally, we thank one and all who directly and indirectly helped us to complete our project successfully.

Project Associates

K. KOSHAL	**(20MH1A4236)**
D. SAI MANOHAR	**(20MH1A4207)**
K. LEELA PRASAD	**(20MH1A4228)**
G. ARUN KUMAR	**(20MH1A4215)**

DECLARATION

This is to declare that the project entitled "**MULTIPLE EYE DISEASE CLASSIFICATION USING DEEP LEARNING**" submitted by us in the partial fulfillment of requirements for the award of the degree of **Bachelor of Technology in CSE - (AI & ML)** in **Aditya College Of Engineering**, is bonafide record of project work carried out by us under the supervision and guidance of **Mr. Y. SURESH KUMAR, M.Tech, Assistant Professor of CSE - (AI & ML)**.

Project Associates

K. KOSHAL	**(20MH1A4236)**
D. SAI MANOHAR	**(20MH1A4207)**
K. LEELA PRASAD	**(20MH1A4228)**
G. ARUN KUMAR	**(20MH1A4215)**

MULTIPLE EYE DISEASE PREDICTION IN OPHTHALIMIC IMAGERY USING DEEP LEARNING

ABSTRACT

This project approach for the simultaneous prediction of multiple eye diseases by leveraging deep learning techniques on ocular images. Our model employs a convolutional neural network (CNN) architecture to analyze and extract intricate patterns from the images, allowing for the accurate identification of various ocular conditions. Through extensive training and validation on diverse datasets, our system demonstrates robust performance in predicting conditions such as glaucoma, diabetic retinopathy, cataract, jaundice and age-related macular degeneration. The results indicate promising capabilities for early detection and intervention, showcasing the potential of deep learning in revolutionizing ophthalmic diagnostics.

This research contributes to the advancement of automated eye disease prediction, paving the way for enhanced clinical decision support systems in the field of ophthalmology. In recent years, deep learning techniques have emerged as powerful tools for medical image analysis, offering the potential to revolutionize the diagnosis and management of various diseases, including those affecting the eyes. In this project, we propose a methodology for the classification of multiple eye diseases using deep learning, with a focus on Convolutional Neural Networks (CNNs).

By leveraging large datasets of retinal images and state-of-the-art deep learning architectures, such as CNNs, we aim to develop an accurate and efficient system for automated eye disease diagnosis. Through the integration of advanced image preprocessing techniques, model training, and deployment into a user-friendly web interface, our project seeks to enable clinicians to rapidly and accurately identify and classify a wide range of eye conditions, including diabetic retinopathy, glaucoma, age-related macular degeneration (AMD), and more.

INDEX

1. INTRODUCTION

This study investigates the utilization of profound learning in ophthalmology for prescient examination, expecting to empower early location and proactive administration of different eye sicknesses. By utilizing Convolutional Brain Organizations (CNNs) and breaking down various imaging modalities including retinal sweeps, fundus pictures, Optical Intelligence Tomography (OCT) examines, and Electronic Wellbeing Records (EHRs), the review delineates patients in light oftheir gamble profiles. This approach gives urgent prognostic bits of knowledge to customized treatment methodologies.

Besides, the joining of multimodal imaging information and longitudinal patient records upgrades the prescient abilities of the models. This works with opportune references, empowers successful sickness checking, and improves asset designationinside medical care frameworks. At last, this proactive methodology adds to working on quiet results and easing the weight on medical care frameworks.
Generally, this study proclaims a promising change in outlook in ophthalmic consideration, stressing the conservation of vision and improvement of the personal satisfaction for people impacted by eye illnesses.

Deep learning-based classification systems have demonstrated remarkable success in various medical imaging tasks, including tumor detection, pathology identification, and disease prognosis. Leveraging the hierarchical representation learning capabilities of CNNs, these systems can extract discriminative features from raw image data and classify them into different disease categories with high accuracy and efficiency. In the context of eye diseases, deep learning offers the potential to streamline the diagnostic process, facilitate early detection, and improve patient outcomes.

In this project, we aim to develop a deep learning-based system for the classification of multiple eye diseases, leveraging large datasets of retinal images and advanced deep learning architectures. By integrating state-of-the-art image preprocessing techniques, model training methodologies, and deployment strategies, our project seeks to create a user-friendly and scalable solution for automated eye disease diagnosis. Through rigorous evaluation and validation, we aim to demonstrate the effectiveness and reliability of our classification system in real-world clinical settings, ultimately contributing to the advancement of ophthalmology and healthcare as a whole.

2. LITERATURE SURVEY

“Deep Learning for Ocular Disease Recognition: An Inner-Class Balance,”

M. S. Khan, N. Tafshir, K. N. Alam et al., Hindawi Computational Intelligence and Neuroscience, Volume 2022. The main focus of the paper was to explore the use of deep learning specifically the vgg-19 model, for the classification and recognition of ocular diseases the study aimed to address the imbalance in the dataset and improve the accuracy of disease prediction using convolutional neural network and transfer learningtechniques.

“Eye Disease Classification Using Deep Learning Techniques”,

Tareq Babaqi, Manar Jaradat, Ayse Erdem Yildirim, Saif H. Al-Nimer, and Daehan Won Department of Systems Science & Industrial Engineering State University of New York at Binghamton, Binghamton, NY, 13902.
Using eye pictures, numerous research has attempted to anticipate and/or categorize a healthy eye from one with a disease presenting a robust automated method based on a proprietary CenterNet model and a Dense Net-100 feature extractor for detecting and classifying diabetic retinopathy and diabetic macular edema lesions. They attained accuracies of 97.93% and 98,10%, respectively, while evaluating their methodology using the APTOS-2019 and IDRID benchmark datasets. Provided three classification techniques for multiple classes: Convolutional Neural Network (CNN), Visual Geometry Group 16 (VGG16), and Inception V3. Using a confusion matrix, the precision of each method is measuredand compared.

“Automatic detection of 39 fundus diseases and conditions in retinal photographs using deep neural networks”,

This study was supported by the National Natural Science Foundation of China (NSFC, 81570849 to L.-P.C., 81800822 to S.T.), Natural Science Foundation of Guangdong

Province, China (NSFG, 2020A1515010415 to L.-P.C.), Grant for Key Disciplinary Project of Clinical Medicine under the Guangdong High-level University Development Program (002-18119101), LKSF cross-disciplinary research grants (2020LKSFG16B to L.P.C. and M.Z.), and an internal grant from the Joint Shantou International Eye Center of The Shantou University and The Chinese University of Hong Kong.
Retinal fundus diseases can lead to irreversible visual impairment without timelydiagnoses. and appropriate treatments Single disease-based deep learning algorithms had been developed for the detection of diabetic retinopathy, age-related macular degeneration, and glaucoma. Here, we developed a deep learningplatform (DLP) capable of detecting multiple common referable fundus diseases and conditions (39 classes) by using 249,620 fundus images marked with 275,543 labels from heterogeneous sources. Our DLF achieved a frequency-weightedaverage Fl score of 0.923, sensitivity of 0.978, specificity of 0.996 and area under the receiver operating characteristic curve (AUC) of 0.9984 for multi-label classification in the primary test dataset and reached the average level of retina specialists. External multihospital tests, public data tests, and tele-reading applications also showed high efficiency for multiple retinal diseases and conditions detection. These results indicate that our DLP can be applied for retinalfundus disease triage, especially in remote areas around the world.

“Multi-Class Retinal Diseases Detection Using Deep CNN With Minimal Memory Consumption”,

Asif Nawaz Muhammad Babar 1, Tariq Ali 1, Ghulammustafa 2,(Member, IEEE),And Basit Qureshi 3,(Senior Member, IEEE), published On:1 June 2023. Machine Learning (ML) such as Artificial Neural Network (ANN), Deep learning, Recurrent Neural Networks (RNN), Alex Net, and ResNet can be considered as abroad research direction in the identification and classification of critical diseases, CNN and its particular variant, usually named U-Net Segmentation, has made a revolutionary advancement in the classification of medical diseases, specifically retinal diseases. However, because of the feature extraction complexity, U-Net has a significant flaw in high memory and CPU consumption while moving the whole feature map to the corresponding decoder Furthermore, it can be concatenated to the unsampled decoder feature map to avoid reusing pooling indices In this research work, a convolutional neural network (CNN) model is proposed for multi-class classification problems with the efficient use of memory consumption. The proposed model has been evaluated on a standard benchmark dataset of Eye Net, having 32 classes of retinal diseases. From experimental evaluation, it has been concluded that the proposed model performs better regarding meniny management and accuracy. The overall comparison has been performed based on precision, recall, and accuracy with different members of epochs and time consumption by each step. The proposed technique achieved an accuracy of 95% on the Eye-net dataset.

“ Deep Learning Fundus Image Analysis for Diabetic Retinopathy and Macular Edema Grading”,

Jaakko Sahisten, Joel Jaskari, Jyri Kivinen, Lauri Turunen, Esa Jaanio, Kustaa Hietala & Kimmo Kaski. Published on: 24 July 2019. Diabetes is a globally prevalent disease that can cause visible microvascular complications such as diabetic retinopathy and macular edema in the human eye retina, the images of which are today used for manual disease screening and diagnosis. This labor-intensive task could greatly benefit from automatic detection using deep learning techniques. Here we present a deep learning system that identifies referable diabetic retinopathy comparably or better than presented in the previous studies, although we use only a small fraction of images (1/6) in training but are aided with higher image resolutions. We also provide novel results for five different screening and clinical grading systems for diabetic retinopathy and macular edema classification, including state-of-the-art results for accuratelyclassifying images according to clinical five-grade diabetic retinopathy and d for f the first time f for the four-grade diabetic macular edema scales. These results suggest, that a deep learning system could increase the cost-effectiveness of screening and diagnosis, while attaining higher than recommended performance, and that the system could be applied in clinical examinations requiring finer grading.

Ocular Disease Recognition using Deep Learning,

Kuldeep Vayadande, Varad Ingale, Abhishek yeoli. Published on: 2022
Artificial intelligence holds a significant impact in a variety of drug-related medical studies, including ophthalmology. Deep literacy styles, in particular, have been successful in detecting clinical signs and bracketing optical conditions. Studies reveal Ocular diseases to be the major contributing reason of childhood blindness all over the world. Rapid and automatic illness identification is vital and urgent in lowering the strain of ophthalmologists. Ophthalmologists use pattern recognition to identify disorders by looking at the eye and its surrounding tissuesdirectly or indirectly. As a result, can benefit the area of medical greatly. Each disease has several severity levels that can be identified by confirming the presence of different lesions. Morphological characteristics identify each lesion, and numerous lesions from different diseases have similar characteristics. In ophthalmology, deep literacy techniques have mostly been employed on eye fundus pictures and optic consonance tomography. In this paper, we have used three models namely CNN, Inception V3, and VGG-19 for cataract prediction. We have got an accuracy of 0.9587 for VGG-19 which is performing best as compared to other models.

A newly developed CNN Model for Eye Disease Detection,

S.A.Toki, S.Rahman, S.M.Billah Fahim, Published on: 2022 2nd International Mobile, Intelligent, and UbiquitousComputing Conference (MIUCC). Fundus images are commonly used by medical experts like ophthalmologists, which are very helpful in detecting various retinal disorders. They used this to diagnose the different types of eye diseases like Cataracts, Diabetic Retinopathy,Glaucoma etc. These fundus images can be also used for the prediction of the severity of the diseases and can provide early signs or warnings. Recently, different machine learning algorithms are playing a vital role in the field of medical science, and it is no different in Ophthalmology either. In this research, we aim to automatically classify healthy and diseased retinal fundus images using deep neural networks. Because deep learning is an excellent machine learning algorithm, which has proven to be very accurate in computer vision problems. Inour research, we used convolutional neural networks (CNN) to classify the retinalimages whether they are healthy or not.

Multiple eye disease detection using Deep Neural Network,

Krishna Prasad, Sajith P S, Neema M. Published on:2019. India has a blind population of approximately 15 million, and the sad reality is that75% of these cases are curable. The doctor-patient ratio in India is 1:10,000. Studieshave found that Diabetic Retinopathy(DR) and Glaucoma are the leading causes ofblindness in India. Diabetic retinopathy is caused mainly due to enduring diabetes in a person and is found to be the principal cause of blindness among the working-age population in developed and developing nations. Glaucoma causes damage tothe optic nerve, thereby leads to blindness. Early stages of both diseases are asymptomatic, making its detection difficult, and if untreated, this can cause irreversible damage to vision. The proposed deep neural network model helps to detect the presence of diabetic retinopathy and glaucoma at its early stages. It can alert the patients to consult an ophthalmologist from a screening standpoint. The developed model is less complicated and resulted in an accuracy of 80%.

An Improved Approach for Detection of Diabetic Retinopathy Using Feature Importance and Machine Learning Algorithms,

S M Asiful Huda, Ishrat Jahan Ila, Shahrier Sarder, Md. Shamsujjoha, Md.Nawab Yousuf Ali. Published on: 2019 7th International Conference on Smart Computing & Communications (ICSCC). Diabetic Retinopathy is a human eye disease that causes damage to the eye's retina and may ultimately result in complete blindness. Early detection of diabetic retinopathy is needed to avoid complete blindness. Physical tests, such as visual acuity test, dilation of pupils, and optical consistency tomography, is used to detect diabetic retinopathy. However, it is costly in terms of time and might affect the patients. In these consequences, this paper detects the presence of Diabetic Retinopathy in the human eye using machine learning algorithm. The proposedmethod applies classification algorithms on several features (e.g., optical disk diameter, lesion-specific (microaneurysms, exudates) or presence of hemorrhages) of an existing Diabetic Retinopathy dataset. Then the features were extracted and used final decision making to predict the presence of diabetic retinopathy. The proposed system used Decision Tree, Logistic Regression. Support Vector Machine for the prediction. The proposed method achieved 88% accurate results which is much better than the existing works. Moreover, the proposed method achieves a better score in precision and recall which are 97% and 92%, respectively compared to the existing result 72% and 63%, i.e., more the 25% in each category on average which proves the enormousness of the proposed method.

Disease Prediction based on Retinal Images,

S. N. Shivappriya, Harikumar Rajaguru, M. Ramya. Published on: 2021 Smart Technologies, Communication and Robotics (STCR). The paper's major goal is to create a basic framework for recognising disorders including diabetes, hypertension, and heart attacks. In the retinal fundus image shows two types of blood vessels that is arteries and veins. To diagnosis the different types of diseases, it is more important to distinguish the vessels into arteries & veins, which are highly correlated with brain and heart functioning. Preprocessing steps are introduced for the segmentation vessel using median filter algorithm. After pre-processing the images are undergone with Convolution Neural Network (CNN) approach. CNN is used to recognize the key feature of the images and they are differentiated based on the key feature. When we give theinput of retinal images the difference in the retinal eye was observed based on their diseases.

There are numerous diseases like heart attack, diabetes, etc, whichkill people. With the help of this project, some of the diseases are predicted with the help of the retinal eye. The retinal eye is very important which consists of retinal nerves which are connected to the brain. The VGG 19 architecture is usedto find the diseases using retinal images.

3. PROBLEM DEFINITION

3.1-Keywords:

Multiple eye diseases – prediction – ocular images – deep learning – glaucoma –diabetic retinopathy – cataract – efficientnet – resnet – VGG – confusion matrix– parameters – accuracy – loss – validation loss – training loss – validation accuracy – recall – precision – epochs

3.2-Deep Learning:

Deep Learning (DL) is a class of Artificial Intelligence (AI) methods inspired bythe structure of the human brain and is based on artificial neural networks. Essentially, DL refers to methods of learning the mathematical representation of the latent and intrinsic relations of the data in an automatic manner. Unlike traditional machine learning methods, deep learning ones require much less human guidance since they are not based on the generation of hand-crafted features, a task that can be very laborious and time-consuming, but instead learn appropriate features directlyfrom the data. In addition, DL methods scale much better than traditional ML methods as the amount of data increases. In this section, a short overview of somekey DL concepts is provided.

3.3-Cataract:

Eyes are the unique vision gland in the human body and many people are suffering from eye disorders that cause vision impairments.

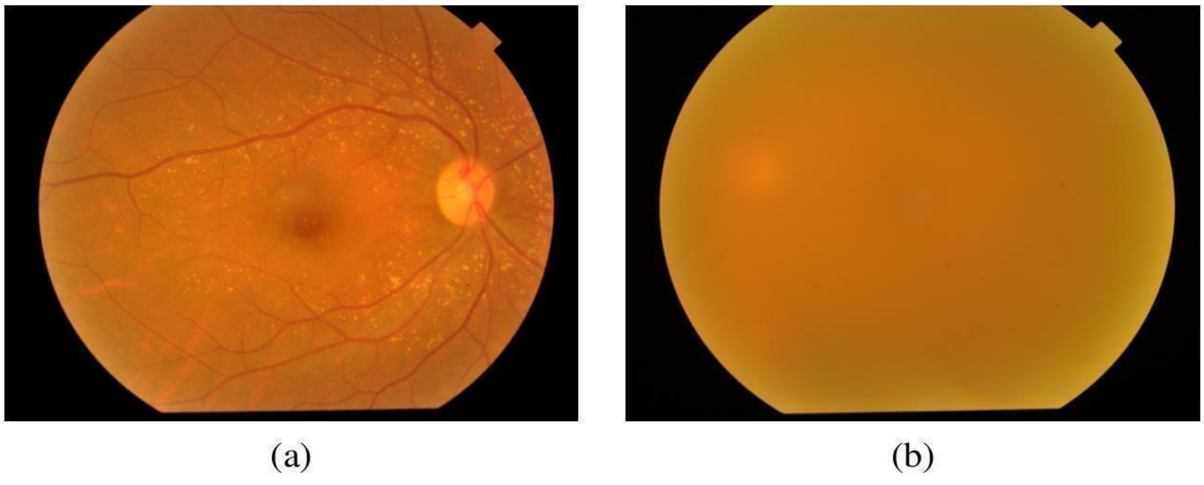


Fig. 1. Retinal Fundus Image (a) non-cataract (b) cataract

Cataract is one of the most prevalent eye diseases which is the frequent reason for blindness. Fig. 1 shows a retinal fundus image which represents the normal retinal fundus image that visible the capillaries and vascular cells (Fig. 1a). Fig. 1b shows a cataract image in which capillaries and vascular are not visible due to blurriness. At thisstage, vision is lost in most of the people.

3.4-Glaucoma:

Glaucoma is the main reason of visual disability across the world and it has no cure. At an early stage, if it is not detected then it can be the cause of permanent blindness. There are cures to prevent vision loss if it is recognizedat an early stage. Since it is a salient chronic eye disease that develops permanent blindness.

Thus, it is important to do eye screening for detecting of glaucoma. The eyescreening process

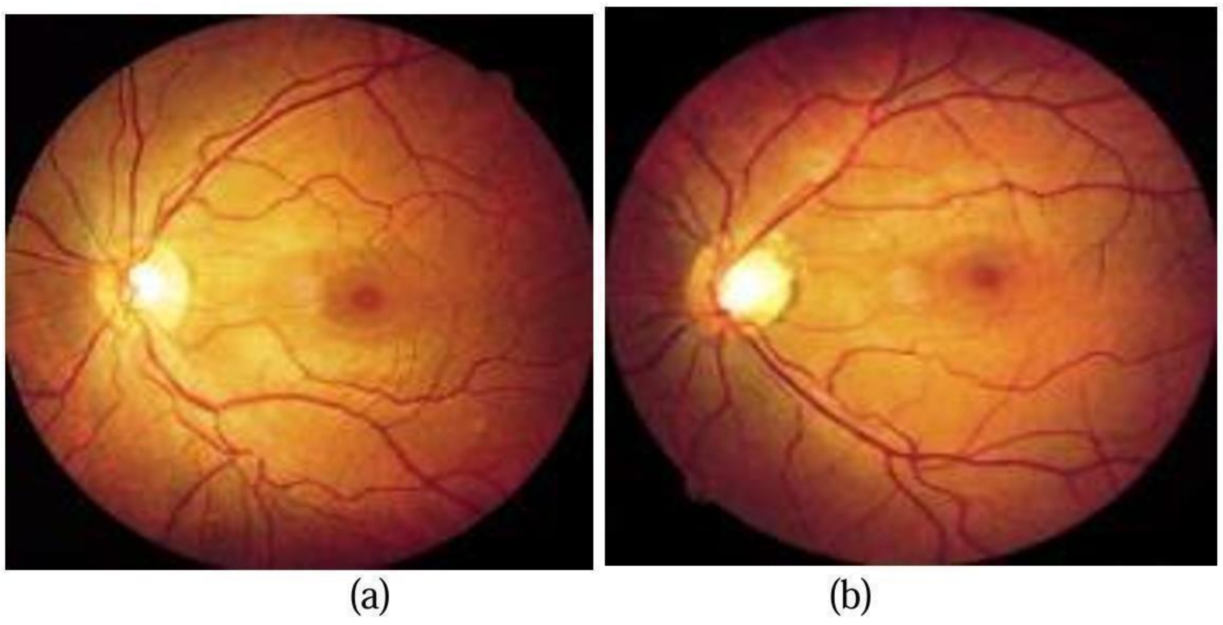


Fig. 1. An example of visually presented glaucoma: (a) normal retinal image, (b) glaucoma.

is expected to tedious and time-consuming task due to the checkup of every individual patient, which is generally large. For ophthalmologists to do better eye screening processes, computer-aided diagnostics systems (CADx) are developed to provide a cost-effective solution to the patients. As a result, the automated CADx screening systemscan reduce time and effort wasted on the analysis of glaucomaeye disease.

It is also important to develop CADX systems using image analysis for clinical experts to differentiate between normal and glaucoma retinal images because it is difficult for ophthalmologists to make this discrimination as shown in Fig. As shown in this figure, the CUP boundary is affected due to glaucoma eye disease.This effect can be determined through the cup-to-disc ratio (CDR). According to theliterature survey, the classification of retinal fundus images into normal and glaucomatous stages is a significant problem due to the large population of screening. There is another indicator for glaucoma eye disease that is known as peripapillary atrophy (PPA). In the case of PPA effect, a change in intensity adjoining the disc boundary can be observed clearly. Fig. 2 depicts the effect of PPA on the glaucomatous retinal fundus region-of-interest (ROI) image. It can be observed from this figure that the PPA appears as a variation in the disc boundary and is similar to rim thinning.

3.5-Diabetic Retinopathy:

Diabetic Retinopathy (DR) is a harmful disease and the main cause of blindness among the working-age population. Moreover, DR is the most feared complication of diabetes and increases the chance of the onset of other diseases,such as kidney disorders, heart disease, and mortality. The onset and progressionof DR are most significantly associated with three risk factors: an increase in blood pressure, poor glycemic control, and long periods in a diabetic condition. The figure shows the eye structure of a healthy person and that of a DR patient.

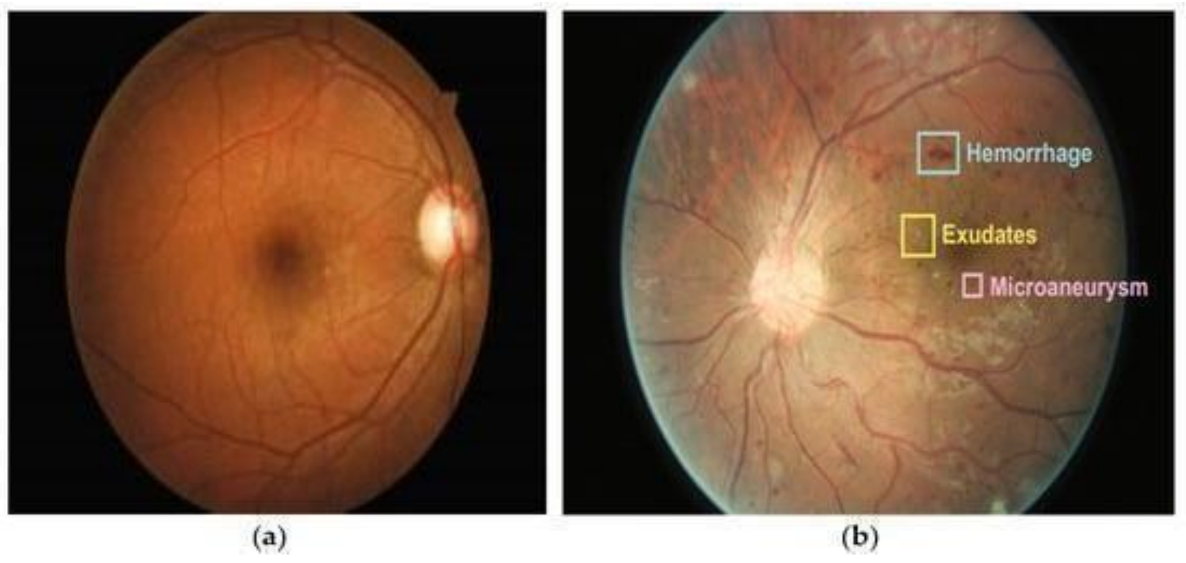


Figure 1. (**a**) Eye structure of Non-DR patient; (**b**) Eye structure of DR patient

4. EXISTING METHODOLOGY

Before the advent of deep learning, traditional machine learning Algorithms, and image-processing techniques were commonly used for eye disease classification. Here's an overview of an old methodology for multiple eye disease classification:

4.1-Image Preprocessing:

Image Acquisition: Collect retinal images using fundus cameras or other imaging devices.

Normalization: Adjust brightness, contrast, and color balance to ensure consistency across images.

Image Enhancement: Techniques like histogram equalization, contrast stretching, and noise reduction to improve image quality.

Segmentation: Identify regions of interest (e.g., optic disc, bloodvessels) using techniques like thresholding, edge detection, or clustering.

4.2-Feature Extraction:

Extract relevant features from preprocessed images to representdifferent characteristics of eye diseases. Handcrafted features could include texture features (e.g., Haralick texture features), shape features, color features, etc. Features could also be extracted from specific regions of interest, such as the optic disc or lesions.

4.3-Feature Selection:

Use techniques like principal component analysis (PCA), linear discriminant analysis (LDA), or feature ranking methods to select the most discriminative features. Reduce dimensionality and remove irrelevant or redundant features to improve classification performance.

4.4-Classification:

Train machine learning classifiers using the selected features. Commonly used classifiers include Support Vector Machines (SVM), Random Forests, k-nearest Neighbors (k-NN), and Naive Bayes. Ensemble methods like AdaBoost or Gradient Boosting could also be employed to improve performance. Implement one-vs-all or one-vs-one strategies for multi-class classification.

4.5-Model Evaluation:

Evaluate the trained models using performance metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC). Employ cross-validation techniques to ensure robustness and generalization of the model.

4.6-Post-Processing:

Post-processing steps may include refining classification results, such as removing false positives or merging overlapping regions.

4.7-Deployment:

Once the model is trained and evaluated, deploy it in a clinical setting or integrate it into a software application for practical use.

4.8-Limitations:

- Handcrafted feature extraction relies heavily on domain knowledge and may not capture complex patterns effectively.
- Performance highly depends on the quality and relevance of the extracted features.
- Difficulty in handling large and high-dimensional image data efficiently.
- Limited scalability and adaptability compared to deep learning approaches.

5. PROPOSED METHODOLOGY

5.1-Convolutional Neural Networks (CNNs):

CNNs are a class of deep neural networks specifically designed for processing structured grid-like data, such as images. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. CNNs leverage the concept of local receptive fields and weight sharing to extract hierarchical features from input images.
The convolutional layers apply filters (kernels) across the input image to detect features like edges, textures, and patterns. Pooling layers downsample the feature maps obtained from convolutional layers to reduce spatial dimensions while preserving important features. Fully connected layers at the end of the network perform high-level feature aggregation and classification.

5.2-VGG19:

VGG19 is a variant of the VGG (Visual Geometry Group) network architecture proposed by the Visual Geometry Group at the University of Oxford. It consists of 19 layers, including 16 convolutional layers and 3 fully connected layers. VGG19 is known for its simplicity and uniform architecture, with small 3x3 convolutional filters and max-pooling layers. The use of small filters allows the network to learn more complex features while keeping the number of parameters manageable. VGG19 achieved excellent performance on the ImageNet dataset, demonstrating its effectiveness in image classification tasks. While VGG19 has a high computational cost due to its depth and number of parameters, it can be fine-tuned or used as a feature extractor for transfer learning.

5.3-Multiple Eye Disease Classification using CNNs and VGG19:

To classify multiple eye diseases using CNNs or VGG19, you would first preprocess your retinal image dataset and label each image with the corresponding disease(s). Then, you would train the CNN or VGG19 model on the preprocessed dataset, adjusting hyperparameters and optimizing the architecture as needed. During training, the model learns to extract relevant features from retinal images and classify them into different disease categories. Transfer learning can be applied by initializing the CNN or VGG19 with pre-trained weights on large-scale image datasets like ImageNet, which can improve training efficiency and performance, especially with limited data. After training, you would evaluate the model's performance using metrics such as accuracy, precision, recall, and F1-score on a separate test dataset. The trained model can then be deployed into a production environment, such as a web application, where users can upload retinal images, and the model predicts the presence and severity of multiple eye diseases.

Advantages:

- CNNs and VGG19 are well-suited for image classification tasks due to their ability to automatically learn hierarchical features from raw pixel data.
- Transfer learning with pre-trained models like VGG19 allows leveraging features learned from large-scale datasets, reducing the need for large annotated datasets and training time.
- These models can achieve state-of-the-art performance in multiple eye disease classification, contributing to early detection and effective management of eye conditions.

5.4-EfficientNet:

EfficientNet is a convolutional neural network architecture and scaling methodthat uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrarily scales thesefactors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. For example, suppose we want to use 2^N times more computational resources. In that case, we can increase the network depth by α^N, width by β^N, and image size by γ^N where α, β, γ, are constant coefficients determined by a small grid search on the original small model. EfficientNet uses a compound coefficient Φ to uniformly scale network width, depth, and resolution in a principled way.

The compound scaling method is justified by the intuition that if the input imageis bigger, then the network needs more layers to increase the receptive field andmore channels to capture more fine-grained patterns on the bigger image.

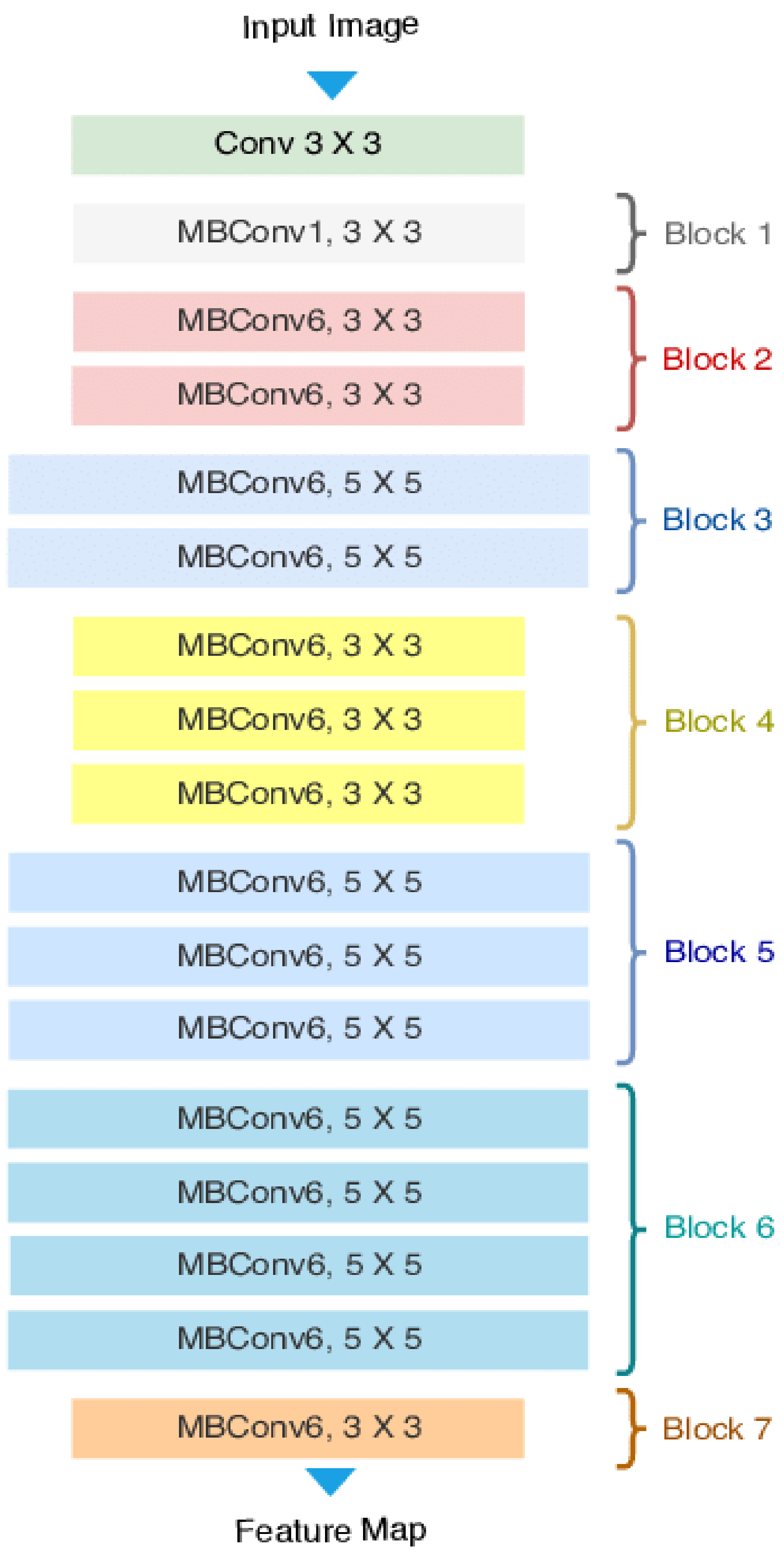
The base EfficientNet-B0 network is based on the inverted bottleneck residual blocks of MobileNetV2, in addition to squeeze-and-excitation blocks.

EfficientNets also transfer well and achieve state-of-the-art accuracy on CIFAR-100 (91.7%), Flowers (98.8%), and 3 other transfer learning datasets, with an order of magnitude fewer parameters. EfficientNet is a convolutional neural network built upon a concept called "compound scaling.” This concept addresses the longstanding trade-off betweenmodel size, accuracy, and computational efficiency. The idea behind compoundscaling is to scale three essential dimensions of a neural network: width, depth, and resolution.

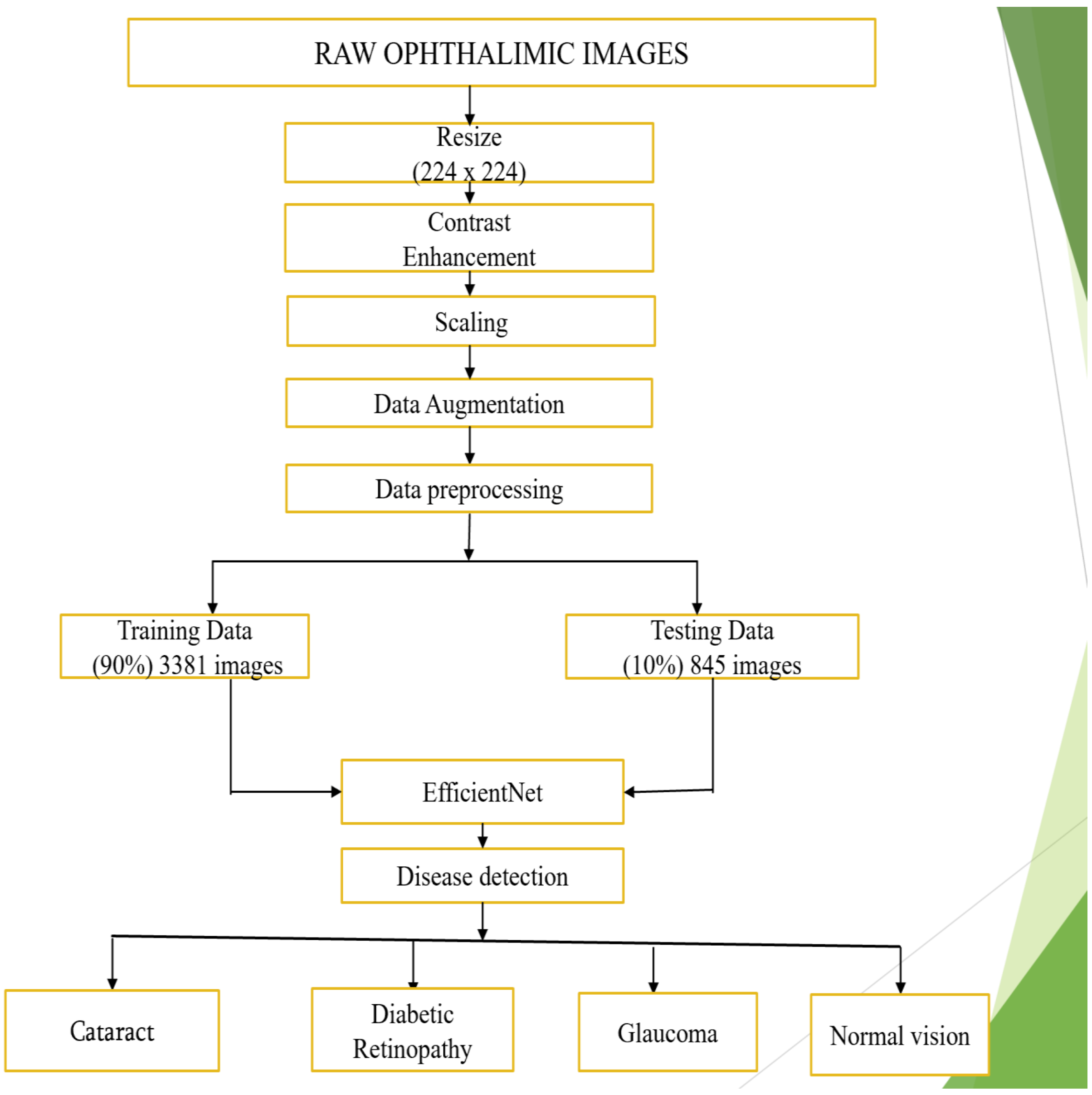
Width: Width scaling refers to the number of channels in each layer of the neural network. By increasing the width, the model can capture more complex patternsand features, resulting in improved accuracy. Conversely, reducing the width leads to a more lightweight model, suitable for low-resource environments.

Depth: Depth scaling pertains to the total number of layers in the network. Deeper models can capture more intricate representations of data, but they also demand more computational resources. On the other hand, shallower models are computationally efficient but may sacrifice accuracy.

Resolution: Resolution scaling involves adjusting the input image's size. Higher- resolution images provide more detailed information, potentially leading to better performance. However, they also require more memory and computational power. Lower-resolution images, on the other hand, consume fewer resources but may lead to a loss in fine-grained details. EfficientNet uses Mobile Inverted Bottleneck (MBConv) layers, which are a combination of depth-wise separable convolutions and inverted residual blocks. Additionally, the model architecture uses the Squeeze-and-Excitation (SE) optimization to further enhance the model’s performance.


Input Image
Conv 3 X 3
MBConv1, 3 X 3
Block 1
MBConv6, 3 X 3
MBConv6, 3 X 3
Block 2
MBConv6, 5 X 5
MBConv6, 5 X 5
Block 3
MBConv6, 3 X 3
MBConv6, 3 X 3
MBConv6, 3 X 3
Block 4
MBConv6, 5 X 5
MBConv6, 5 X 5
MBConv6, 5 X 5
Block 5
MBConv6, 5 X 5
MBConv6, 5 X 5
MBConv6, 5 X 5
MBConv6, 5 X 5
Block 6
MBConv6, 3 X 3
Block 7
Feature Map

5.5-Proposed Methodology:



- **Step-1:** The raw ophthalmic images are undergone for preprocessing. In Data Pre-Processing the images are cropped to 224x224 and the image contrast andbrightness are enhanced.

- **Step-2:** After Data Pre-Processing the images are classified as Train & Testimages which are 90% and 10% respectively.

- **Step-3:** The training dataset is undergone for data augmentation.

- **Step-4:** Now the training dataset is undergone for training in different neuralnetworks such as simple EfficientNet.

- **Step-5:** Finally these neural networks are compared and selected the best neural network for ophthalmic diseases detection by observing the test accuracies, f1scores, precisions, and Confusion matrices of every neural network. This entire process is represented as a flow chart.

6. SYSTEM REQUIREMENTS

- **6.1-Software Requirements**

Compiler: Python compiler 3.8.5

Software used: Anaconda, Visual Studio Code, Jupyter Notebook, Google Colab

Deep Learning Framework: TensorFlow, Keras

Frontend: HTML, CSS, JS

Backend: Django, Flask

Image Processing Libraries: OpenCV

- **6.2-Hardware Requirements**

OS: Windows 10 or Higher, Mac OS or higher, 64-bit, Linux

Ram: 8 GB

Disk Space requirement: 10 GB

Graphic Card: NVIDIA GEFORCE GTX 1050ti

7. SYSTEM DESIGN

Software design is a process of problem-solving and planning for software solutions after the purpose and specification of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low level component and algorithm implementation issues as well as the architectural view.

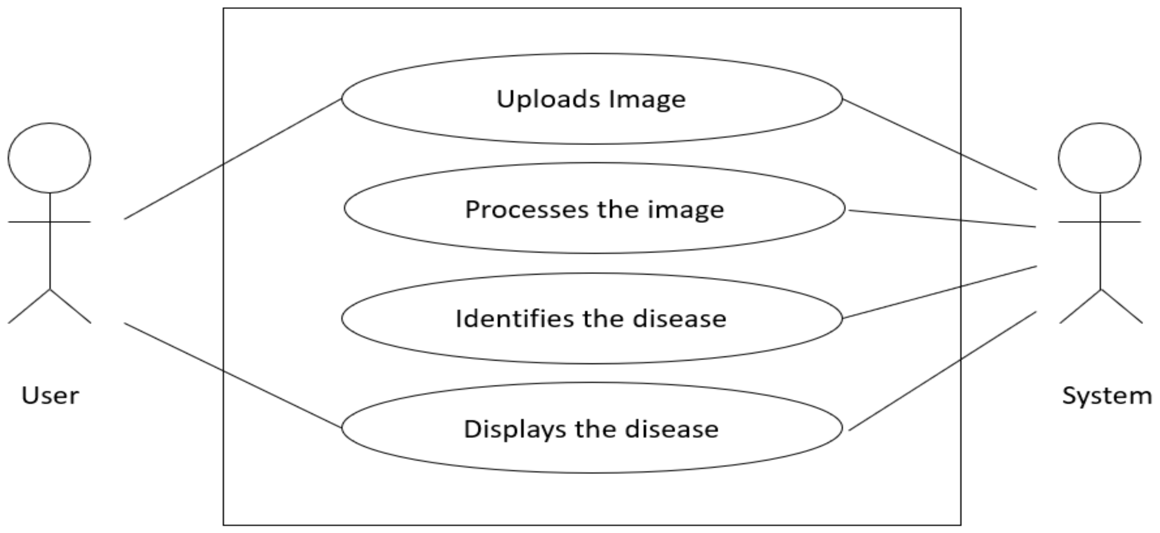
UML Diagrams

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML gives us a standard way to write systems blueprints, covering conceptual things, such as business processes and system functions, as well as classes written in a specific programming language, database schemas, and reusable software components.

7.1-Use Case Diagram

Use Case diagrams are one of the five diagrams in UML for modeling the dynamic aspects of systems. Use Case diagrams are central to modeling the behavior of a system, a subsystem, or a class. Actors are external entities that interact with the system. Examples of actors include users like clients, administrators, etc, or another system like a central database.

The overview of the whole project through use case diagram is as follows:

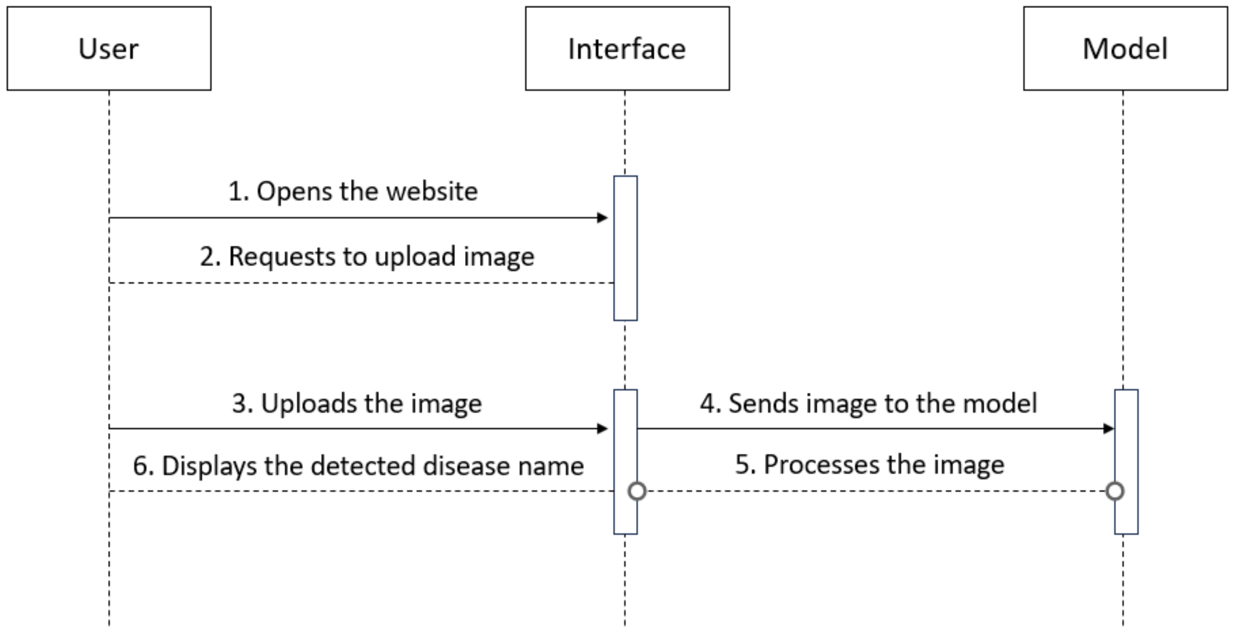


The user initiates the process by selecting the desired photo of eye from their local device and uploading it to the system. Once the upload is complete, the website's backend system begins processing the image. Utilizing advanced image processing and deep learning algorithms, the system analyzes the image and identifies the disease that is associated with the eye. Upon successful eye disease classification, the system gives output as the disease name and is then presented to the user through the website's interface. The user can view the disease name.

7.2-Sequence Diagram

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram in a table that shows objects arranged along the X axis and messages, ordered in increasing time along the Y axis. It contains the object life line that represents the existence of an object over a period of time. There is a focus on control that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

The overview of the whole project through a sequence diagram is as follows:

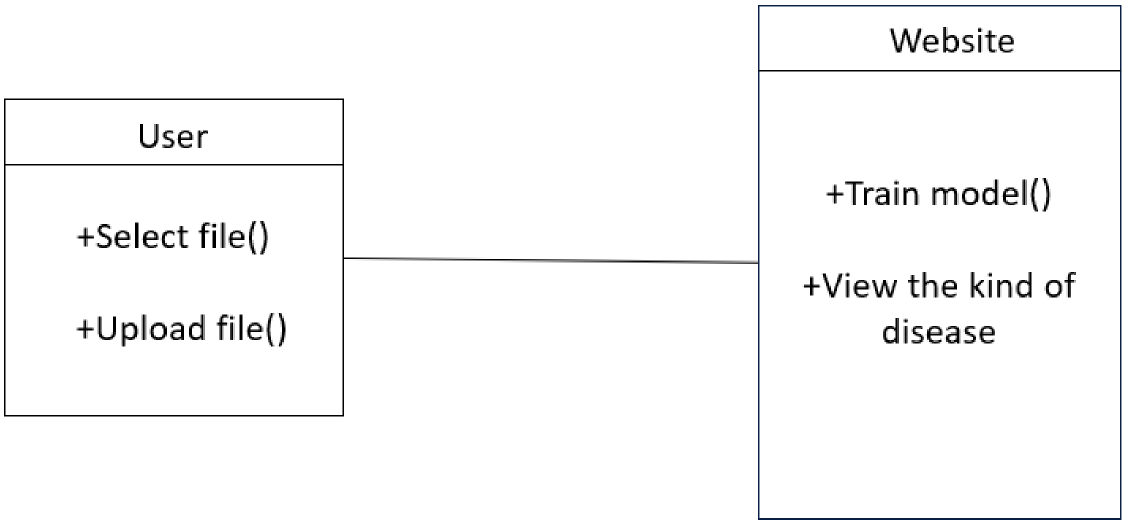

User
Interface
Model
1. Opens the website
2. Requests to upload image
3. Uploads the image
4. Sends image to the model
6. Displays the detected disease name
5. Processes the image

The user initiates the process by selecting the desired photo of eye from their local device and uploading it to the system. Once the upload is complete, the website's backend system begins processing the image. Utilizing advanced image processing and deep learning algorithms, the system analyzes the image and identifies the disease that is associated with the eye. Upon successful eye disease classification, the system gives output as the disease name and is then presented to the user through the website's interface. The user can view the disease name.

7.3-Class Diagram:

A class diagram is a type of UML (Unified Modeling Language) diagram used to visualize the structure of a system by showing the classes, their attributes, methods, and relationships between them. In a class diagram, classes are represented as boxes with three compartments: the top compartment contains the class name, the middle compartment lists the class attributes, and the bottom compartment displays the class methods. Relationships between classes are depicted using lines with various annotations to indicate the type of relationship, such as association, aggregation, or inheritance. Class diagrams are an essential tool for software developers during the design phase of a project, providing a high-level overview of the system's architecture and helping to communicate design decisions effectively among team members. They serve as a blueprint for implementation, guiding developers in writing code that adheres to the defined structure and relationships specified in the diagram.

The overview of the whole project through the class diagram is as follows:



8. SYSTEM TESTING

8.1-Introduction

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

8.2-Psychology of Testing

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn’t work. Testing is the process of executing a program with the intent of finding the errors.

8.3-Testing Objectives

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test uncovers an as-yet-undiscovered error.
- A good test case has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

8.4-Types of Testing

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

8.4.1-Unit Testing

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

8.4.2-Integration Testing

After the unit testing, we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

8.4.3-System Testing

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if the software meets its requirements. Here entire project has been tested against the requirements of the project and it is checked whether all requirements of the project have been satisfied or not.

8.4.4-Acceptance Testing

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behaviour of the system; the internal logic of program is not
emphasized.

9. CODING AND PLOTTING RESULTS

9.1-Coding:

Import necessary modules:

```
!pip install tensorflow==2.9.1# import system libs
import os import time import shutil import pathlib import itertools

# import data handling toolsimport cv2
import numpy as npimport pandas as pd
import seaborn as sns sns.set_style('darkgrid')import matplotlib.pyplot as plt
from sklearn.model_selectionimport train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# import Deep learning Libraries import tensorflow as tffrom tensorflow import keras
from tensorflow.keras.models import Sequential

from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator



from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation,
Dropout, BatchNormalization
from tensorflow.keras import regularizers

# Ignore Warningsimport warnings
warnings.filterwarnings("ignore")print ('modules loaded')
```

Function to create data frame

```
# Generate data paths with labelsdef define_paths(data_dir):
os.listdir(data_dir)
for fold in folds:
foldpath = os.path.join(data_dir, fold)filelist = os.listdir(foldpath)

for file in file list:
fpath = os.path.join(foldpath, file)filepaths.append(fpath) labels.append(fold)
return filepaths, labels

# Concatenate data paths with labels into one dataframe ( to later be fitted intoThe model )
def define_df(files, classes):
Fseries = pd.Series(files, name= 'filepaths') Lseries = pd.Series(classes,
name = ‘labels’ )
return pd.concat([Fseries, Lseries], axis= 1)
# Split dataframe to train, valid, and testdef split_data(data_dir):
# train dataframe
files, classes = define_paths(data_dir)

df = define_df(files, classes) strat = df['labels']
train_df, dummy_df = train_test_split(df, train_size= 0.8, shuffle= True,rand om_state= 123,
stratify= strat)

df = define_df(files, classes)strat = df['labels']
train_df, dummy_df = train_test_split(df, train_size= 0.8, shuffle= True, random_state= 123,
stratify= strat)

# valid and test dataframe strat = dummy_df['labels']
valid_df, test_df = train_test_split(dummy_df, train_size= 0.5, shuffle= True,random_state=
```

123, stratify= strat)
return train_df, valid_df, test_df

Function to generate images from dataframe
def create_gens (train_df, valid_df, test_df, batch_size):
''' This function takes train, validation, and test dataframe and fit them into the image data generator because model takes data from the image data generator.
Image data generator converts images into tensors. '''

# define model parameters img_size = (224, 224) channels = 3 # either BGR or Grayscale color = 'rgb'img_shape = (img_size[0], img_size[1], channels)

Recommended: use a custom function for test data batch size, else we can use normal batch size.
ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length + 1) ifts_length%n == 0 and ts_length/n <= 80]))
test_steps = ts_length // test_batch_size

# This function which will be used in image data generator for data augmentation, it just take the image and return it again.

def scalar(img): return imgreturn img
tr_gen = ImageDataGenerator(preprocessing_function= scalar, horizontal_flip=True)
ts_gen = ImageDataGenerator(preprocessing_function= scalar)

train_gen = tr_gen.flow_from_dataframe(train_df, x_col= 'filepaths', y_col= ' l labels', target_size= img_size, class_mode= 'categorical', color_mode= color,
huffle= True, batch_size= batch_size)

```
valid_gen = ts_gen.flow_from_dataframe( valid_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
color_mode= color, shuffle= True, batch_size= batch_size)
# Note: we will use custom test_batch_size, and make shuffle= false
test_gen = ts_gen.flow_from_dataframe( test_df, x_col= 'filepaths', y_col= 'la bels', target_size= img_size, class_mode= 'categorical',
color_mode= color, shuffle= False, batch_size= test_batch_size)
return train_gen, valid_gen, test_gen Function to display data sampledef show_images(gen):
This function take the data generator and shows sample of the images '''
# return classes , images to be displayed
g_dict = gen.class_indices # defines dictionary {'class': index}
classes = list(g_dict.keys()) # defines list of dictionary's kays (classes), classesnames : string
images, labels = next(gen)# get a batch size samples from the generator

# calculate number of displayed samples length = len(labels)      # length of batch size
sample = min(length, 25) # check if sample less than 25 imagesplt.figure(figsize= (20, 20))

for i in range(sample): plt.subplot(5, 5, i + 1)
image = images[i] / 255   # scales data to range (0 - 255)plt.imshow(image)
index = np.argmax(labels[i]) # get image index class_name = classes[index] # get class of imageplt.title(class_name, color= 'blue', fontsize= 12) plt.axis('off')
plt.show
```

Examples: stop model training after specfic time, stop training if no improvement inaccuracy and so on.

Linkcode

```
class MyCallback(keras.callbacks.Callback):
def___init___(self, model, patience, stop_patience, threshold, factor, batches, e pochs,
```

ask_epoch):

super(MyCallback, self)._init___() self.model = model

self.patience = patience # specifies how many epochs without improvemen tbefore learning rate is adjusted

self.stop_patience = stop_patience # specifies how many times to adjust lr without improvement to stop training

self.threshold = threshold # specifies training accuracy threshold when lr w illbe adjusted based on validation loss

self.factor = factor # factor by which to reduce the learning rate self.batches =batches # number of training batch to run per epoch self.epochs = epochs self.ask_epoch = ask_epoch self.ask_epoch_initial = ask_epoch # save this value to restore if restartingtraining

# callback variables

self.count = 0 # how many times lr has been reduced without improvement self.stop_count = 0

self.best_epoch = 1 # epoch with the lowest loss

self.initial_lr = float(tf.keras.backend.get_value(model.optimizer.lr)) # get theinitial learning rate and save it

self.highest_tracc = 0.0 # set highest training accuracy to 0 initially self.lowest_vloss = np.inf # set lowest validation loss to infinity initially self.best_weights = self.model.get_weights() # set best weights to model's initial weights

self.initial_weights = self.model.get_weights() # save initial weights if the yhave to get restored

# Define a function that will run when training begins

def on_train_begin(self, logs= None):

msg = 'Do you want model asks you to halt the training [y/n] ?'print(msg)

ns = input('')

if ans in ['Y', 'y']: self.ask_permission = 1 elif ans in ['N', 'n']: self.ask_permission = 0

msg = '{0:^8s}{1:^10s}{2:^9s}{3:^9s}{4:^9s}{5:^9s}{6:^9s}{7:^10s}{8: 10s}{9:^8s}'.format('Epoch', 'Loss', 'Accuracy', 'V_loss', 'V_acc', 'LR', 'Next LR ', 'Monitor','% Improv', 'Duration')

print(msg)
self.start_time = time.time()

def on_train_end(self, logs=None):stop_time = time.time()
tr_duration = stop_time - self.start_time hours = tr_duration // 3600minutes = (tr_duration - (hours * 3600)) // 60
seconds = tr_duration - ((hours * 3600) + (minutes * 60))
msg = f’training elapsed time was {str(hours)} hours, {minutes:4.1f} minutes, {seconds:4.2f} seconds)'print(msg)

# set the weights of the model to the best weightsself.model.set_weights(self.best_weights)

def on_train_batch_end(self, batch, logs= None): # get batch accuracy and loss acc = logs.get('accuracy') * 100 loss = logs.get('loss')

# prints over on the same line to show running batch count
msg = '{0:20s}processing batch {1:} of {2:5s}-accuracy= {3:5.3f}-

loss: {4:8.5f}'.format(' ', str(batch), str(self.batches), acc, loss)print(msg, '\r', end= '')

def on_epoch_begin(self, epoch, logs= None):

self.ep_start = time.time()

# Define method runs on the end of each epoch def on_epoch_end(self, epoch,logs= None):

ep_end = time.time()

duration = ep_end - self.ep_startself.ep_start = time.time()

    self.ep_start = time.time()

# Define method runs on the end of each epoch def on_epoch_end(self, epoch,logs= None):

ep_end = time.time()

duration = ep_end - self.ep_start

lr = float(tf.keras.backend.get_value(self.model.optimizer.lr)) # get the curr ent learning rate current_lr = lr

acc = logs.get('accuracy') # get training accuracy

v_acc = logs.get('val_accuracy') # get validation accuracy loss = logs.get('loss')# get training loss for this epoch

v_loss = logs.get('val_loss') # get the validation loss for this epoch

if acc < self.threshold: # if training accuracy is below threshold adjust lr ba sedon training accuracy

monitor = 'accuracy' if epoch == 0:

pimprov = 0.0 else:

pimprov = (acc - self.highest_tracc) * 100 / self.highest_tracc # defin eimprovement of model progres

if acc > self.highest_tracc: # training accuracy improved in the epoch self.highest_tracc = acc # set new highest training accuracy self.best_weights =self.model.get_weights() # training accuracy impr

oved so save the weights

self.count = 0 # set count to 0 since training accuracy improved self.stop_count

= 0 # set stop counter to 0

if v_loss < self.lowest_vloss: self.lowest_vloss = v_loss self.best_epoch = epoch + 1 # set the value of best epoch for this epoch

else:

# training accuracy did not improve check if this has happened for pat iencenumber of epochs # if so adjust learning rate

if self.count >= self.patience - 1: # lr should be adjusted lr = lr * self.factor # adjust the learning by factor tf.keras.backend.set_value(self.model.optimizer.lr, lr) # set the lear ning rate inthe optimizer

self.count = 0 # reset the count to 0

self.stop_count = self.stop_count + 1 # count the number of consec utive lradjustments

self.count = 0 # reset counterif v_loss < self.lowest_vloss:

self.lowest_vloss = v_losselse:

self.count = self.count + 1 # increment patience counter

else: # training accuracy is above threshold so adjust learning rate based onvali dation loss

monitor = 'val_loss'if epoch == 0:

pimprov = 0.0

else:

pimprov = (self.lowest_vloss - v_loss) * 100 / self.lowest_vloss

if v_loss < self.lowest_vloss: # check if the validation loss improved self.lowest_vloss = v_loss # replace lowest validation loss with new validation loss

self.best_weights = self.model.get_weights() # validation loss improv ed sosave the weights

self.count = 0 # reset count since validation loss improved self.stop_count = 0self.best_epoch = epoch + 1 # set the value of the best epoch to this epoch

else: # validation loss did not improve

if self.count >= self.patience - 1: # need to adjust lr lr = lr * self.factor # adjust the

```
learning rate.

self.stop_count = self.stop_count + 1 # increment stop counter beca use lr wasadjusted
self.count = 0 # reset counter tf.keras.backend.set_value(self.model.optimizer.lr, lr) # set the
learning rate in the optimizer
else:
self.count = self.count + 1 # increment the patience counterif acc > self.highest_tracc:
self.highest_tracc = acc
msg = f'{str(epoch + 1):^3s}/{str(self.epochs):4s} {loss:^9.3f}{acc * 100:^
9.3f}{v_loss:^9.5f}{v_acc * 100:^9.3f}{current_lr:^9.5f}{lr:^9.5f}{monitor:^1
1s}{pimprov:^10.2f}{duration:^8.2f}'
print(msg)

if self.stop_count > self.stop_patience - 1: # check if learning rate has been adjusted
stop_count times with no improvement
msg = f' training has been halted at epoch {epoch + 1} after {self.stop_patience} adjustments
of learning rate with no improvement'
print(msg)
self.model.stop_training = True # stop training

else:
if self.ask_epoch != None and self.ask_permission != 0:if epoch + 1 >= self.ask_epoch:
msg = 'enter H to halt training or an integer for number of epochs to run thenask again'
print (msg)
ans=input('')
if ans == 'H' or ans == 'h':
msg = f'training has been halted at epoch {epoch + 1} due to user
print(msg)
self.model.stop_training = True # stop training
```

```
else:
        try:
            ans = int(ans) self.ask_epoch += ans
            msg = f' training will continue until epoch {str(self.ask_epoch)}'



            print(msg)
            msg = '{0:^8s}{1:^10s}{2:^9s}{3:^9s}{4:^9s}{5:^9s}{6:^9s}
            {7:^10s}{8:10s}{9:^8s}'.format('Epoch', 'Loss', 'Accuracy', 'V_loss', 'V_acc', 'L
            R', 'Next LR', 'Monitor', '% Improv', 'Duration')
            print(msg)

            except Exception: print('Invalid')
            #This function take training model and plot history of accuracy and losses with
            the best epoch
            def plot_training(hist):
            tr_acc = hist.history

            al_lowest =    val_loss[index_loss] index_acc = np.argmax(val_acc)
            acc_highest = val_acc[index_acc] Epochs = [i+1 for i in range(len(tr_acc))]
            loss_label = f'best epoch= {str(index_loss + 1)}'acc_label = f'best epoch=
            {str(index_acc + 1)}'plt.figure(figsize= (20, 8)) plt.style.use('fivethirtyeight')
            plt.subplot(1, 2, 1)
```

plt.plot(Epochs, tr_loss, 'r', label= 'Training loss') plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')

plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)plt.title('Training and Validation Loss')

plt.xlabel('Epochs')plt.ylabel('Loss') plt.legend() plt.subplot(1, 2, 2)

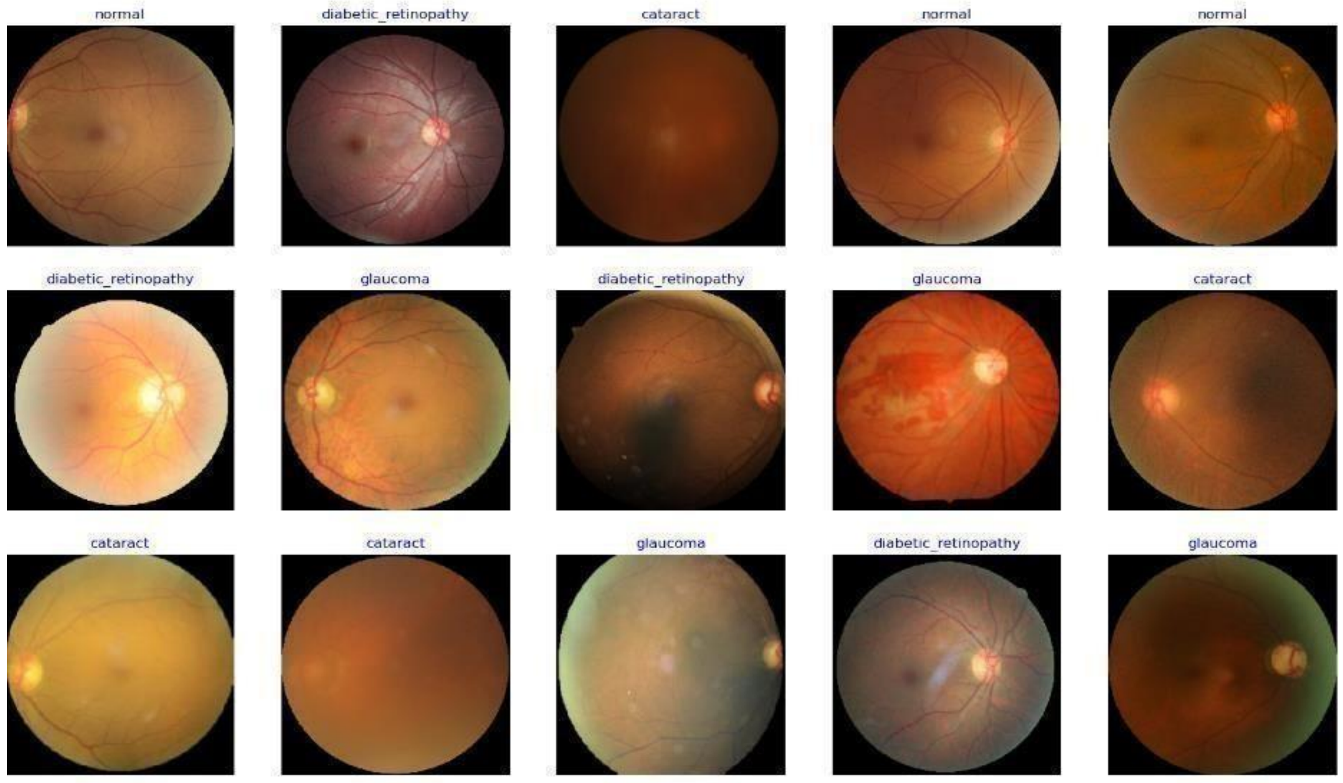
plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy') plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')

plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)plt.title('Training and Validation Accuracy')

plt.xlabel('Epochs') plt.ylabel('Accuracy')plt.legend() plt.tight_layout plt.show()

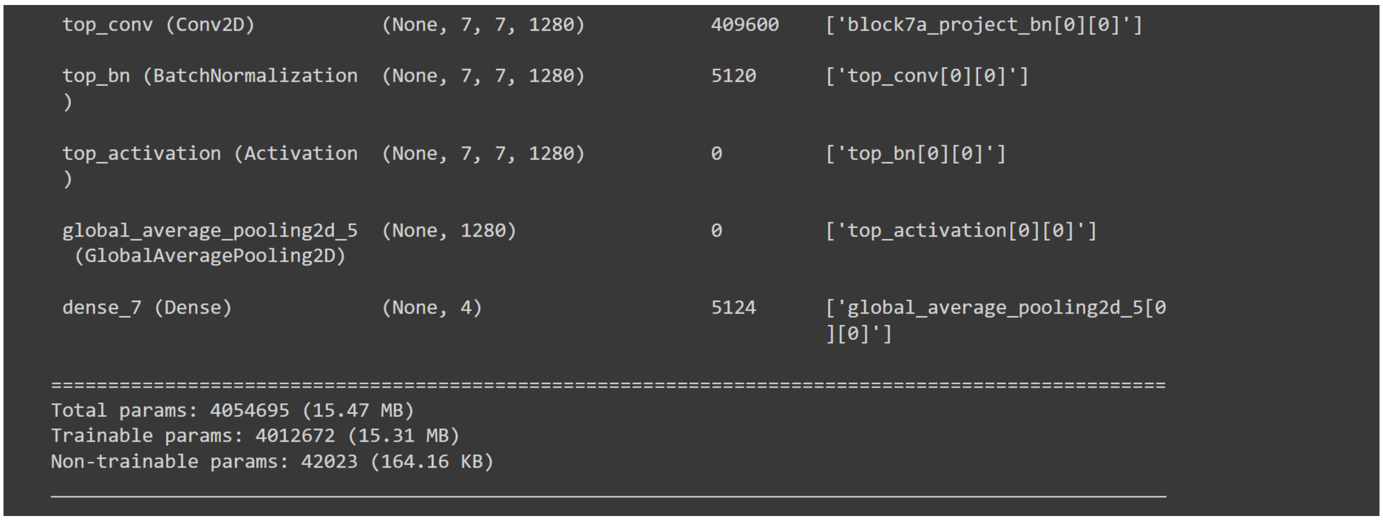
9.2-Display Image Sample:

show_images(train_gen)



Generic Model Creation:

```
# Create Model Structure img_size = (224, 224)channels = 3
img_shape = (img_size[0], img_size[1], channels)

class_count = len(list(train_gen.class_indices.keys())) # to define number of classesin dense
layer

# create pre-trained model (you can built on pretrained model such as : efficientnet,  VGG ,
Resnet )
# we will use efficientnetb3 from EfficientNet family.
base_model = tf.keras.applications.efficientnet.EfficientNetB3(include_top=  Fal se,

eights= "imagenet", input_shape= img_shape, pooling= 'max')

model = Sequential([ base_model,
BatchNormalization(axis= -1, momentum= 0.99, epsilon= 0.001), Dense(256,
kernel_regularizer= regularizers.l2(l= 0.016), activity_regularizer
= regularizers.l1(0.006),

bias_regularizer= regularizers.l1(0.006), activation= 'relu'), Dropout(rate= 0.45,seed= 123),
Dense(class_count, activation= 'softmax')]) model.compile(Adamax(learning_rate= 0.001),
loss= 'categorical_crossentropy',metrics= ['accuracy'])
model.summary()
```



```
 top_conv (Conv2D)           (None, 7, 7, 1280)           409600    ['block7a_project_bn[0][0]']

 top_bn (BatchNormalization  (None, 7, 7, 1280)           5120      ['top_conv[0][0]']
 )

 top_activation (Activation  (None, 7, 7, 1280)           0         ['top_bn[0][0]']
 )

 global_average_pooling2d_5  (None, 1280)                 0         ['top_activation[0][0]']
  (GlobalAveragePooling2D)

 dense_7 (Dense)             (None, 4)                    5124      ['global_average_pooling2d_5[0
                                                                    ][0]']

==================================================================================================
Total params: 4054695 (15.47 MB)
Trainable params: 4012672 (15.31 MB)
Non-trainable params: 42023 (164.16 KB)
__________________________________________________________________________________________________
```

Set Callback Parameters:

batch_size = 40

epochs = 40

patience = 1

stop_patience = 3

threshold = 0.9

factor = 0.5

ask_epoch = 5

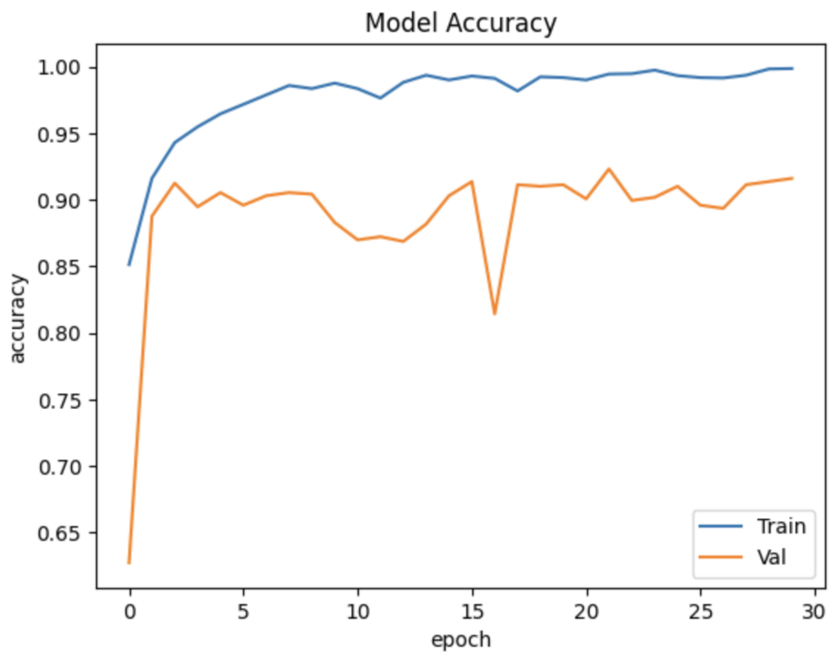
batches = int(np.ceil(len(train_gen.labels) / batch_size))

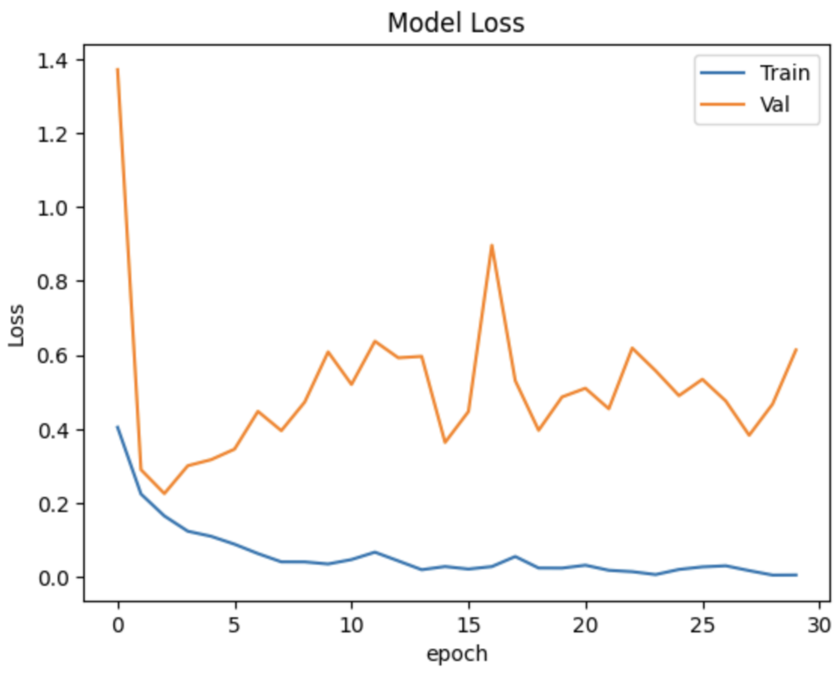
callbacks = [MyCallback(model= model, patience= patience, stop_patience= stop_patience, threshold= threshold, factor= factor, batches= batches, epochs=epochs, ask_epoch= ask_epoch)]

Train Model

history = model.fit(x= train_gen, epochs= epochs, verbose= 0, callbacks= callbacks, validation_data= valid_gen, validation_steps= None, shuffle= False)

9.3-Display model performance:





9.4-Evaluate model:

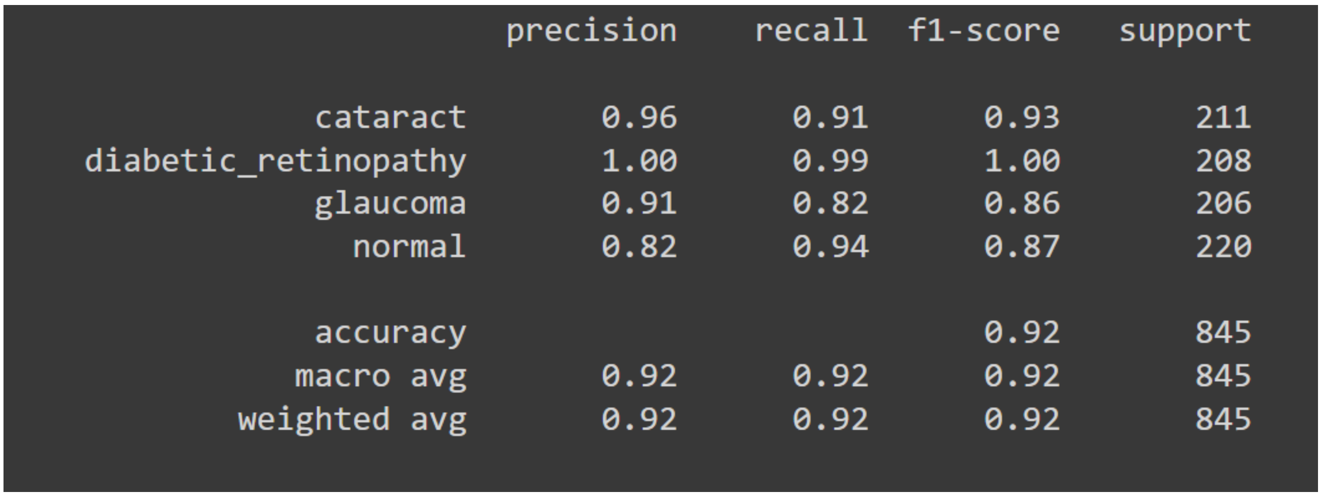
```
ts_length = len(test_df)
test_batch_size = test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length +
1) if ts_length%n == 0 and ts_length/n <= 80]))
test_steps = ts_length // test_batch_size
train_score = model.evaluate(train_gen, steps= test_steps, verbose= 1)
valid_score = model.evaluate(valid_gen, steps= test_steps, verbose= 1) test_score =
model.evaluate(test_gen, steps= test_steps, verbose= 1)
print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1]) print('-' * 20)print("Validation Loss: ",
valid_score[0]) print("Validation Accuracy: ", valid_score[1])
print('-' * 20)
print("Test Loss: ", test_score[0]) print("Test Accuracy: ", test_score[1])
```

9.5-Get predictions:

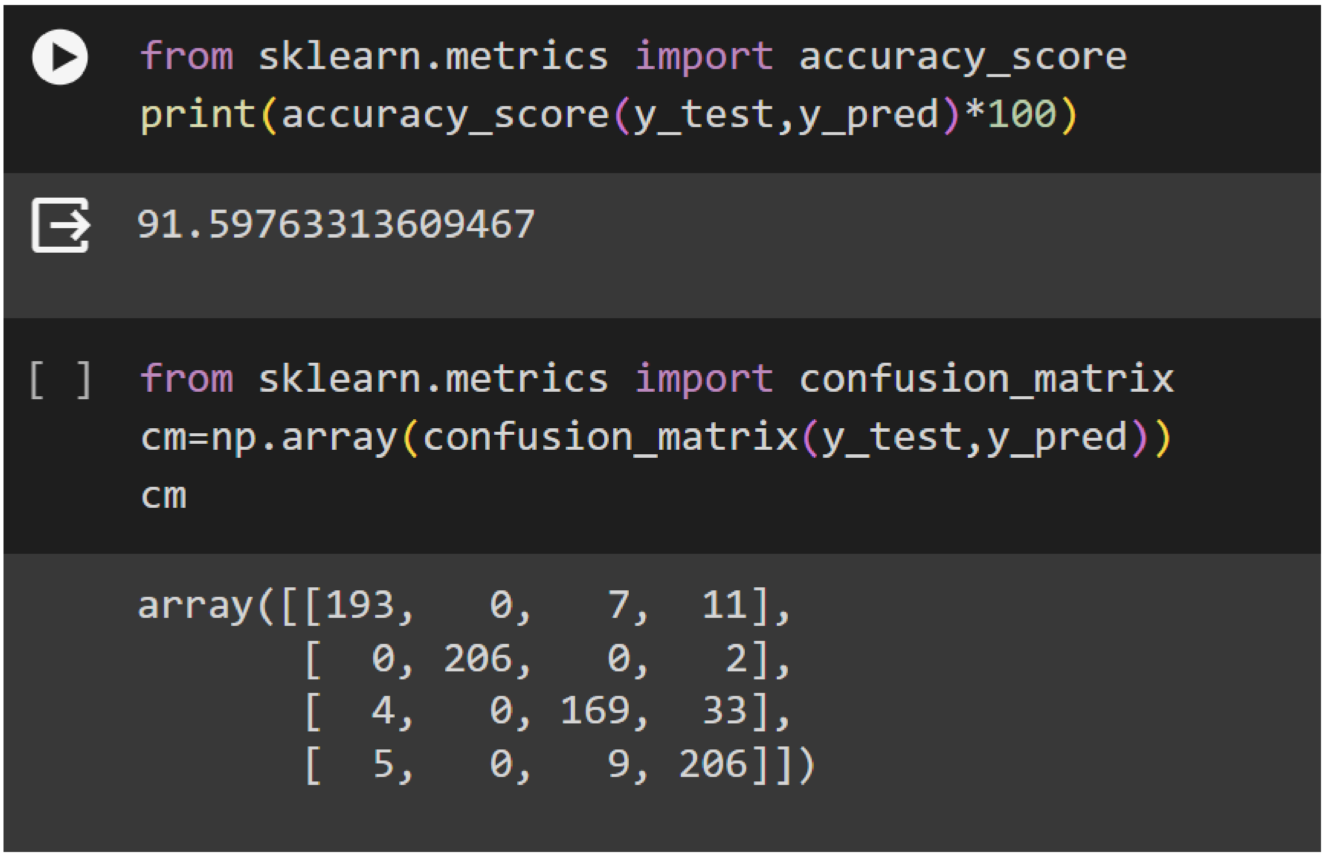
```
preds = model.predict_generator(test_gen)y_pred = np.argmax(preds, axis=1) print(y_pred)
```

Confusion Matrics and Classification Report

```
g_dict = test_gen.class_indices classes = list(g_dict.keys())

# Confusion matrix
cm = confusion_matrix(test_gen.classes, y_pred) plot_confusion_matrix(cm= cm, classes=
classes, title = 'ConfusionMatrix')

# Classification report
print(classification_report(test_gen.classes, y_pred, target_names=classes))
```



```
                      precision    recall  f1-score   support

            cataract       0.96      0.91      0.93       211
diabetic_retinopathy       1.00      0.99      1.00       208
            glaucoma       0.91      0.82      0.86       206
              normal       0.82      0.94      0.87       220

            accuracy                           0.92       845
           macro avg       0.92      0.92      0.92       845
        weighted avg       0.92      0.92      0.92       845
```



```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred)*100)
```

```
91.59763313609467
```

```
[ ]  from sklearn.metrics import confusion_matrix
     cm=np.array(confusion_matrix(y_test,y_pred))
     cm
```

```
array([[193,   0,   7,  11],
       [  0, 206,   0,   2],
       [  4,   0, 169,  33],
       [  5,   0,   9, 206]])
```

```
model_name = model.input_names[0][:-6] subject = 'Eye Disease'acc = test_score[1] * 100
save_path = ''

# Save model
save_id = str(f'{model_name}-{subject}-{"%.2f" %round(acc, 2)}.h5') model_save_loc =
os.path.join(save_path, save_id) model.save(model_save_loc)
print(f'model was saved as {model_save_loc}')

# Save weights
weight_save_id = str(f'{model_name}-{subject}-weights.h5') weights_save_loc =
os.path.join(save_path, weight_save_id) model.save_weights(weights_save_loc)
print(f'weights were saved as {weights_save_loc}')
```

Generate CSV files containing class indices & image size

```
class_dict = train_gen.class_indices img_size = train_gen.image_shape height =[] width = []
for _ in range(len(class_dict)):height.append(img_size[0]) width.append(img_size[1])
Index_series = pd.Series(list(class_dict.values()), name= 'class_index') Class_series =
pd.Series(list(class_dict.keys()), name= 'class')
```

Number of Epochs (s = 30): Each epoch involves one complete pass through the entire training dataset. Training for 30 epochs means the model goes throughthe entire training dataset 30 times. More epochs allow the model to further refine its parameters and potentially improve its performance.

Training Size (training = 3381): Refers to the number of samples in the trainingdataset. With 3381 samples available for training, the EfficientNet model has asubstantial amount of data to learn from. Larger training datasets can providemore diverse examples for the model to learn from, potentially improving itsability to generalize to unseen data.

Testing Size (testing = 845): Indicates the number of samples in the testing dataset used to evaluate the model's performance. With 845 samples available for testing, the model's performance is evaluated on a separate dataset that it hasn't seen during training. Evaluating on a separate testing dataset helps assess the model's ability to generalize to new, unseen data.

Accuracy (91%): Accuracy represents the proportion of correctly classified samples out of the total number of samples in the testing dataset. A model accuracy of 91% indicates that the EfficientNet model correctly classified approximately 91% of the samples in the testing dataset as shown in the confusionmatrix. A higher accuracy indicates better performance in classifying the samples correctly.

In summary, training an EfficientNet-B0 model involves iterating through thetraining dataset for a specified number of epochs, utilizing a large training dataset to learn from, evaluating its performance on a separate testing dataset,and assessing its accuracy as a measure of performance. Achieving a high accuracy indicates that the model has effectively learned to classify the samples in the testing dataset.

DISEASE	PRECISION	RECALL	F1-SCORE	SUPPORT
Cataract	0.94	0.90	0.92	104
Diabetic- retinopathy	1.00	1.00	1.00	110
Glaucoma	0.85	0.87	0.86	101
Normal	0.90	0.91	0.90	107
Macro avg	0.92	0.92	0.92	422
Weighted avg	0.92	0.92	0.92	422

9.6-Parameters:

- **Precision:** Precision is one indicator of t h e machine learning model's performance – the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total numberof positive predictions (i.e., the number of true positives plus the number of false positives).

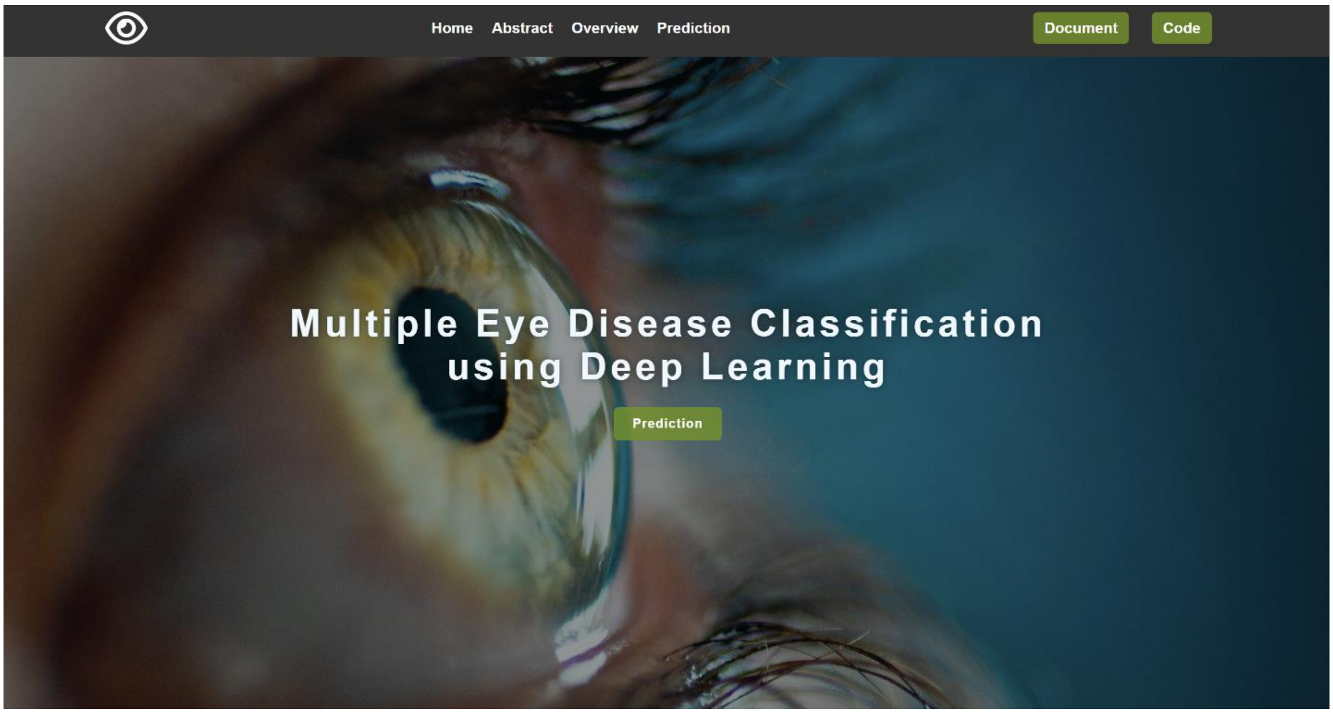
- **Recall:** Recall. also known as the true positive rate (TPR), is the percentage of data samples that a machine learning model correctly identifies as belonging to a class of interest—the "positive class"—out of the total samples for that class.

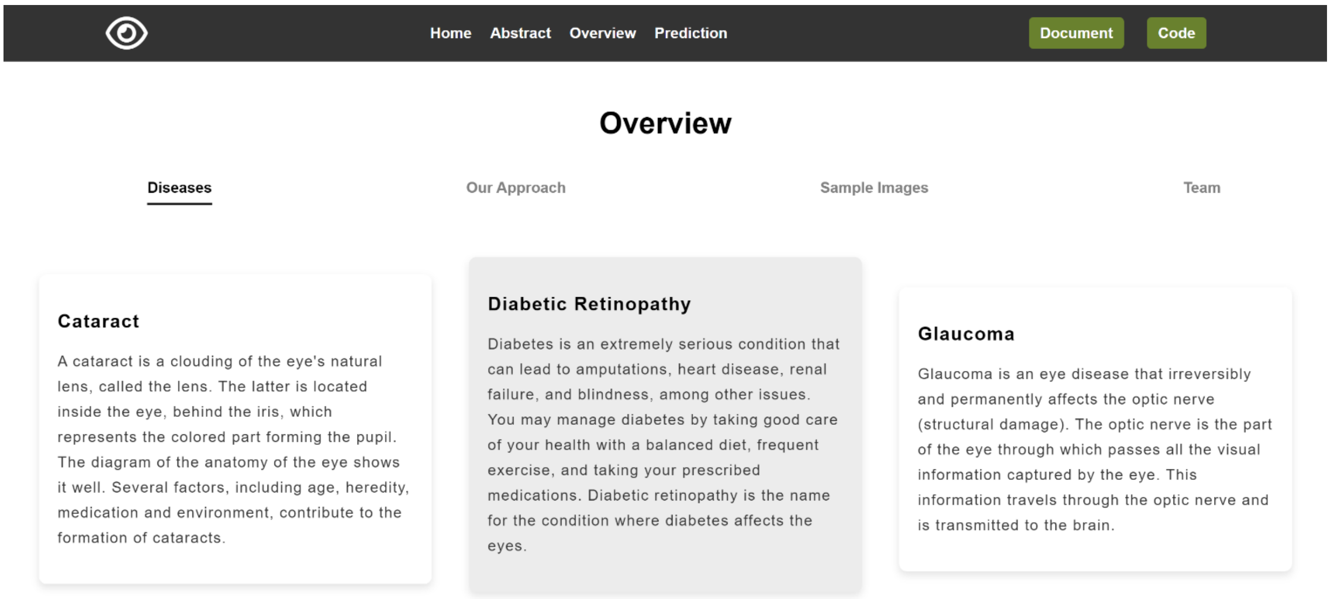
- **Accuracy:** Accuracy is a metric that measures how often a machine learning model correctly predicts the outcome. You can calculate accuracyby dividing the number of correct predictions by the total number of predictions.

- **F1 score:** F1 Score is a measure of the harmonic mean of precision and recall.

- **Epoch:** An epoch is when all the training data is used at once and is defined as the total number of iterations of all the training data in one cycle for training the machine learning model.

9.7-OUTPUT:





Overview

Diseases   Our Approach   Sample Images   Team

Cataract

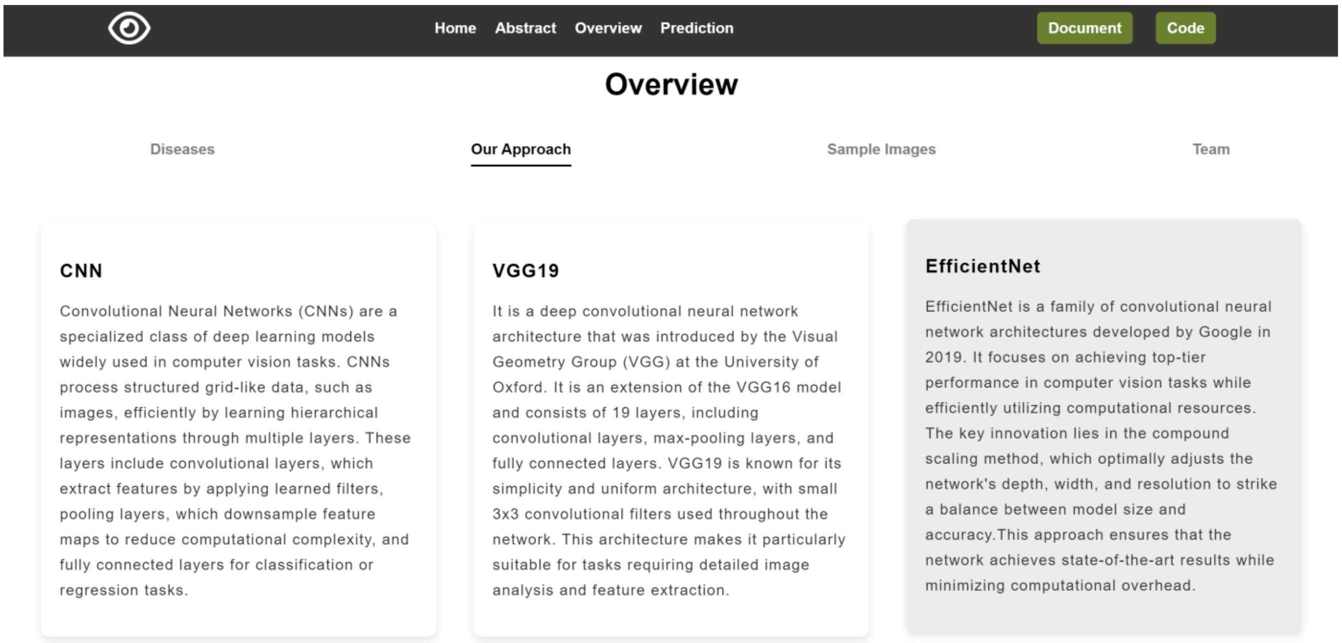
A cataract is a clouding of the eye's natural lens, called the lens. The latter is located inside the eye, behind the iris, which represents the colored part forming the pupil. The diagram of the anatomy of the eye shows it well. Several factors, including age, heredity, medication and environment, contribute to the formation of cataracts.

Diabetic Retinopathy

Diabetes is an extremely serious condition that can lead to amputations, heart disease, renal failure, and blindness, among other issues. You may manage diabetes by taking good care of your health with a balanced diet, frequent exercise, and taking your prescribed medications. Diabetic retinopathy is the name for the condition where diabetes affects the eyes.

Glaucoma

Glaucoma is an eye disease that irreversibly and permanently affects the optic nerve (structural damage). The optic nerve is the part of the eye through which passes all the visual information captured by the eye. This information travels through the optic nerve and is transmitted to the brain.



Overview

Diseases Our Approach Sample Images Team

CNN

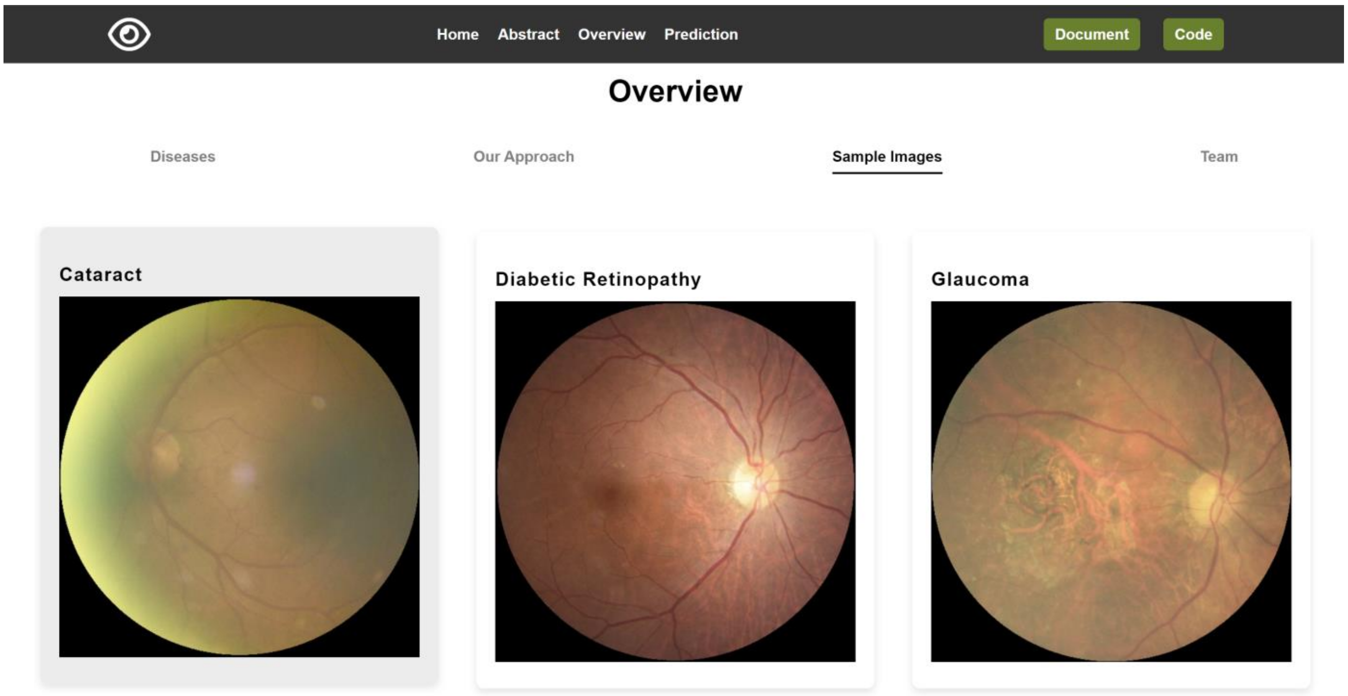
Convolutional Neural Networks (CNNs) are a specialized class of deep learning models widely used in computer vision tasks. CNNs process structured grid-like data, such as images, efficiently by learning hierarchical representations through multiple layers. These layers include convolutional layers, which extract features by applying learned filters, pooling layers, which downsample feature maps to reduce computational complexity, and fully connected layers for classification or regression tasks.

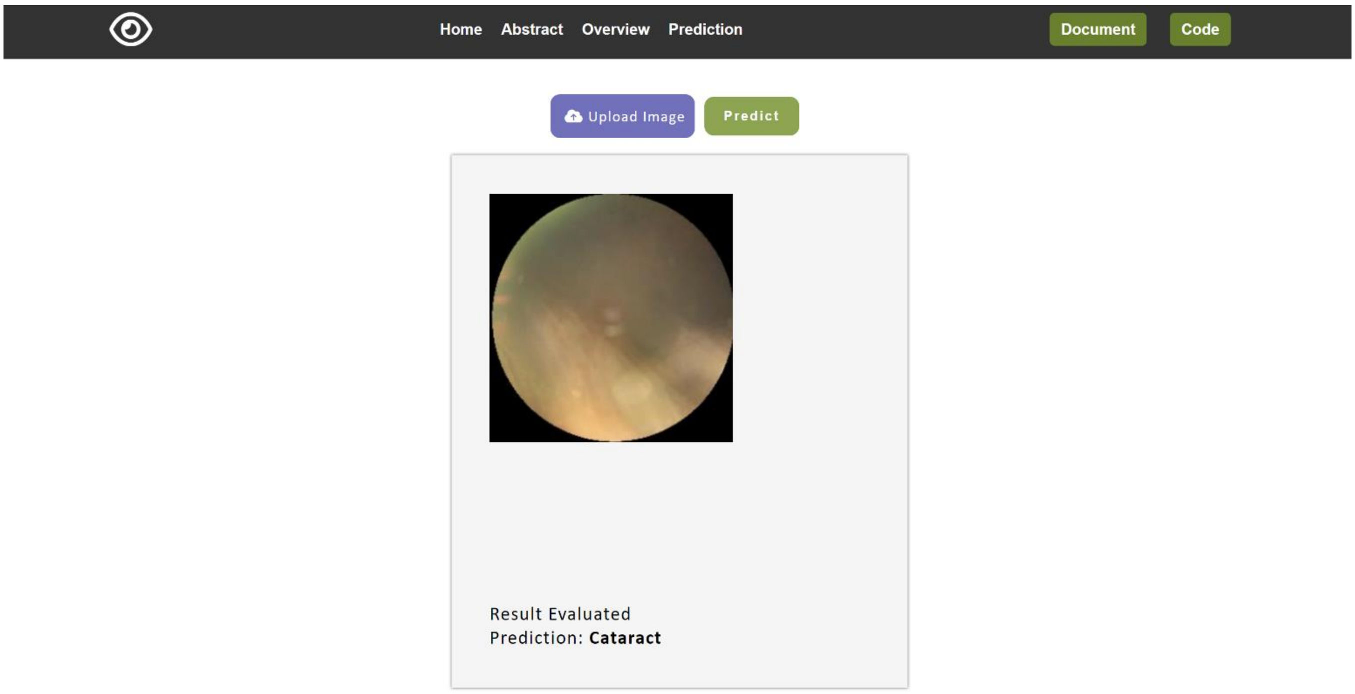
VGG19

It is a deep convolutional neural network architecture that was introduced by the Visual Geometry Group (VGG) at the University of Oxford. It is an extension of the VGG16 model and consists of 19 layers, including convolutional layers, max-pooling layers, and fully connected layers. VGG19 is known for its simplicity and uniform architecture, with small 3x3 convolutional filters used throughout the network. This architecture makes it particularly suitable for tasks requiring detailed image analysis and feature extraction.

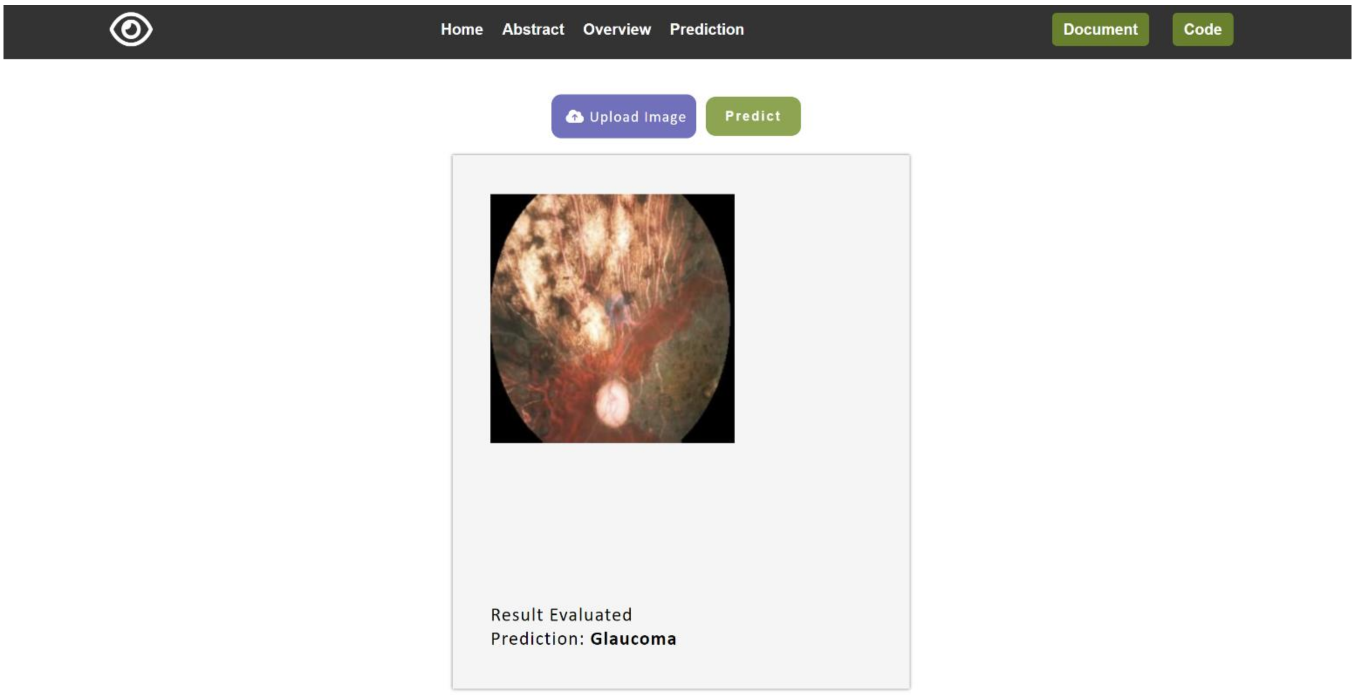
EfficientNet

EfficientNet is a family of convolutional neural network architectures developed by Google in 2019. It focuses on achieving top-tier performance in computer vision tasks while efficiently utilizing computational resources. The key innovation lies in the compound scaling method, which optimally adjusts the network's depth, width, and resolution to strike a balance between model size and accuracy.This approach ensures that the network achieves state-of-the-art results while minimizing computational overhead.

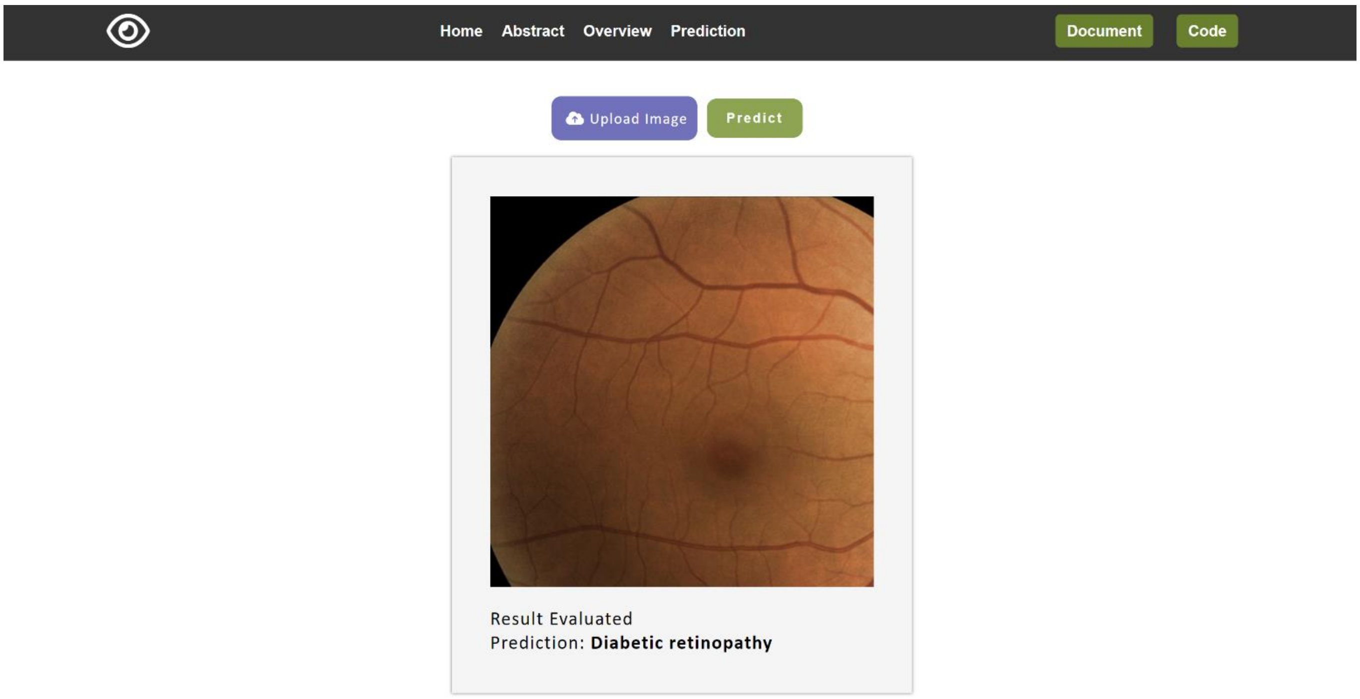




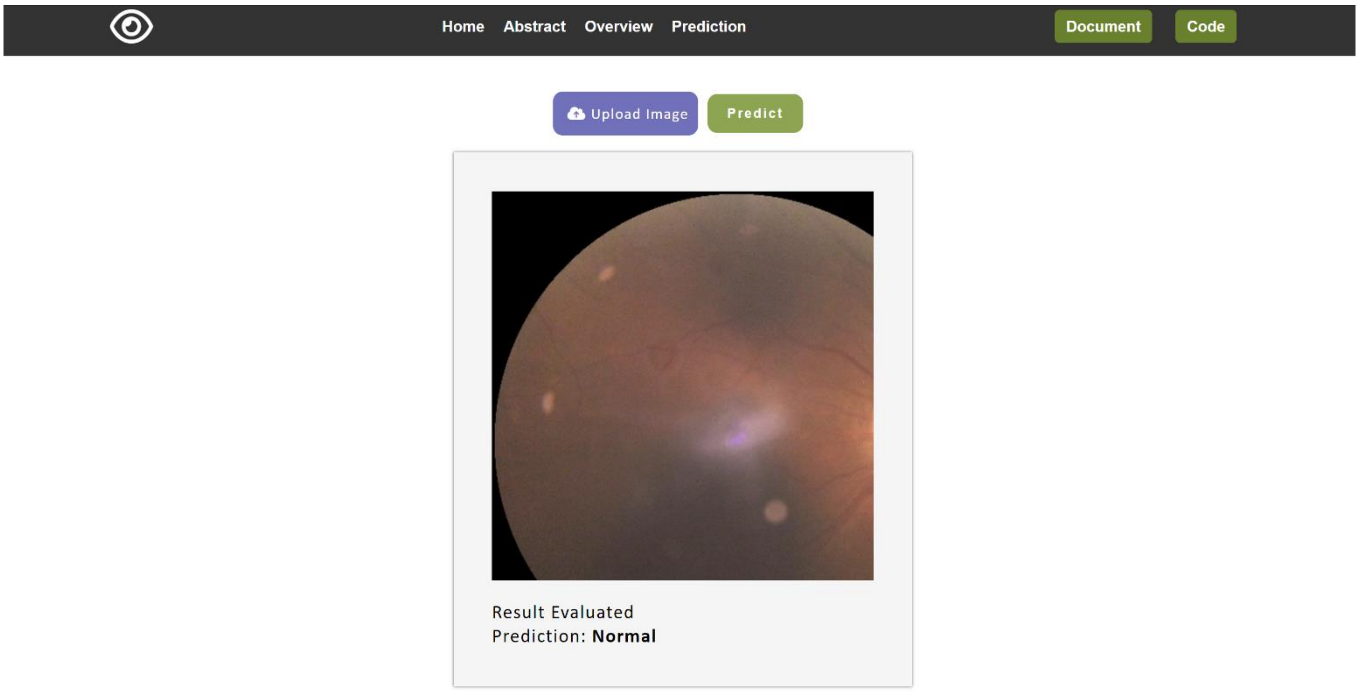
Result Evaluated
Prediction: **Cataract**



Result Evaluated
Prediction: **Glaucoma**



Result Evaluated
Prediction: **Diabetic retinopathy**



Result Evaluated
Prediction: **Normal**

10. CONCLUSION

The eye is one of the most important sense organs in human anatomy, and losingvision would significantly affect the quality of life. Besides, the eye could also indicate some serious health issues. Unfortunately, many people may not be aware of these health problems due to the lack of ophthalmologists and access to eye care. Nevertheless, the advent of deep learning and image processing could immensely help in detecting and diagnosing these diseases. This study used CNN and transfer learning to detect five eye diseases: cataracts, diabetic retinopathy, and glaucoma.

the application of deep learning techniques, particularly Convolutional Neural Networks (CNNs), has demonstrated significant promise in the field of multiple eye disease classification. By leveraging the capabilities of deep learning models, such as feature extraction, hierarchical representation learning, and pattern recognition, researchers and clinicians can enhance the accuracy, efficiency, and accessibility of eye disease diagnosis and management.

Through this project, we have explored various methodologies, including data collection, preprocessing, model selection, training, and deployment, to develop a robust and effective system for classifying multiple eye diseases. By assembling a diverse dataset of retinal images and employing deep learning architectures like CNNs, we can effectively train models to identify and classify various eye conditions, including diabetic retinopathy, glaucoma, age-related macular degeneration (AMD), and more.

The utilization of transfer learning techniques, such as fine-tuning pre-trained models like VGG19, enables us to leverage knowledge learned from large-scale datasets, further enhancing the performance of our classification system, particularly in scenarios with limited annotated data.

The deployment of the trained model into a user-friendly web interface facilitates seamless interaction with clinicians and patients, allowing for rapid and accurate diagnosis and timely intervention. Additionally, continuous monitoring, updates, and refinement of the model ensure its reliability and effectiveness in real-world clinical settings.

- The eye is crucial for human anatomy, and vision loss significantly impacts quality of life.
- Eye health can indicate broader health issues, but many lack access to ophthalmologists and eye care.
- Deep learning and image processing offer potential for detecting and diagnosing eye diseases.
- Study utilizes Convolutional Neural Networks (CNN) and transfers learning to detect five eye diseases: cataracts, diabetic retinopathy, glaucoma, jaundice, and age-related macular degeneration.

11. REFERENCES

- Ocular Disease Recognition using Deep Learning https://ieeexplore.ieee.org/document/10007470
- A newly developed CNN Model for Eye Disease Detection https://ieeexplore.ieee.org/document/9781785
- An Improved Approach for Detection of Diabetic Retinopathy Using FeatureImportance and Machine Learning Algorithms https://ieeexplore.ieee.org/document/8843676
- Multiple eye disease detection using Deep Neural Network https://www.researchgate.net/publication/345354448_Multiple_eye_disease_detection_using_Deep_Neural_Network
- Disease Prediction based on Retinal Images https://ieeexplore.ieee.org/document/9588829
- Deep Learning Fundus Image Analysis for Diabetic Retinopathy and MacularEdema Grading https://www.nature.com/articles/s41598-019-47181-w
- Automatic detection of 39 fundus diseases and conditions in retinalphotographs using deep neural networks https://www.nature.com/articles/s41467-021-25138-w
- Eye Disease Classification Using Deep Learning Techniques https://arxiv.org/abs/2307.10501

- Deep Learning for Ocular Disease Recognition: An InnerClass Balance https://www.hindawi.com/journals/cin/2022/5007111/

- Multi-Class Retinal Diseases Detection Using Deep CNN With Minimal Memory Consumption
  https://ieeexplore.ieee.org/document/10141609