**EyeDeep-Net: a multi-class diagnosis of retinal diseases using deep neural network**

**INTRODUCTION**

Retinal diseases are spreading widely among humans of all ages. The retina contains a layer of optic nerve tissue called photosensitive in the human eye. This layer transforms the light focused by the lens into brain impulses. Macula, positioned in the retina’s middle, performs the sensing process. Information acquired by the macula is processed by the retina and transferred to the brain for visual recognition through the optic nerve. Different types of diseases can cause abnormality in perception, such as age-related macular degeneration (AMD), optic disc drusen, Rothspot diabetic macular edema (DME), etc. In most of the developed countries, people belonging to the age group of 50 to 60 are losing vision due to AMD. According to recent research, in the United States (US), this abnormality is found in about 35% of adults in the age group of 80. Identifying retinal diseases is the most challenging task, as accurate diagnosis needs a highly experienced ophthalmologist due to the diversity of retinal diseases. Similarly, with computer-aided diagnostic systems (CAD), retinal diseases can easily be identified and treated at early stages. Technology advancements have immense benefits in almost every field of life, especially in the medical domain. Several approaches and models have been presented to improve the efficacy and quality of medical solutions. A significant improvement has been observed in the social health system with the advancement in Automatic Disease Detection (ADD). Furthermore, an ADD application, namely retinal symptom analysis, provides a unique opportunity to improve eye care globally. Recently, many state-of-the-art ML and Deep Learning (DL) models have been proposed for the classification, segmentation, and identification of retinal diseases. We observe that data collection and labeling are significant challenges in the implementation of ADDs, as presented by authors in and, due to the development of several machine learning (ML) and deep learning (DL) models, including Recurrent Neural Network (RNN), Convolution Neural Network (CNN), Alex Net ResNet and VGN. These have enabled researchers and physicians to detect and categorize such vital disorders readily. An ML-based Hybrid technique is presented for the classification of retinal diseases automatically. Researchers in have proposed to use U-Net segmentation for image pre-processing; they have also used a Support Vector Machine (SVM) classifier for the classification. The proposed technique achieved a diagnostic accuracy of 89.3%. Yang et al. also provided the first labeled EyeNet dataset containing 32 retinal diseases. It was noted by authors in that the U-Net has a significant flaw of high memory consumption in moving the whole feature map to the corresponding decoder. Deep learning plays a vitol role in the classification of images. This research proposes a CNN model based on deep learning for classifying multi-class eye disease detection. The proposed model has been evaluated on EyeNet Dataset. The EyeNet dataset includes 32 folders, each containing related images for specific. 70% has been used for training and the rest for validation. From experimental evaluation, it has been observed that the proposed model achieved 95% of accuracy. The deep learning-based CNN model has been applied for retinal-based crucial diseases to boost the conventional diagnostic method. This is the primary contribution of this study. The key contributions of the paper are as follows. • A deep learning-based CNN model has been utilized to strengthen the traditional diagnosis process for retinal-based crucial disease. • The proposed CNN model produces better outcomes while consuming low memory than standard state-of-art techniques. • Experimental evaluation reveals that the performance of the proposed model on the multi-class EyeNet dataset produces higher accuracy. The remainder of the paper is structured as follows: Section presents the related works. In Section, we offer the proposed architecture along with a detailed description of the dataset used. Section details the result of the experimental evaluation, including the performance of the given CNN model. Section contains the analysis and the discussion. Finally, section concludes the research work with future directions.

**1.1 Objective of the project:**

Retinal images are a key element for ophthalmologists in diagnosing a variety of eye illnesses. The retina is vulnerable to microvascular changes as a result of many retinal diseases and a variety of research have been done on early diagnosis of medical images to take proper treatment on time. This paper designs an automated deep learning-based non-invasive framework to diagnose multiple eye diseases using colour fundus images. A multi-class eye disease RFMiD dataset was used to develop an efficient diagnostic framework. Multi-class fundus images were extracted from a multi-label dataset and then various augmentation techniques were applied to make the framework robust in real-time. Images were processed according to the network for low computational demand. A multi-layer neural network EyeDeep-Net has been developed to train and test images for diagnosis of various eye problems in which the keystone convolutional neural network extracts relevant features from the input colour fundus image dataset and then processed features were used to make predictive diagnostic decisions. The strength of the EyeDeep-Net is evaluated using multiple statistical parameters and the performance of the proposed model is found to be significantly superior to multiple baseline state-of-the-art models. A comprehensive comparison of the proposed methodology to the most recent methods proves its efficacy in terms of classification and disease identification through digital fundus images.

**2. LITERATURE SURVEY:**

**"Dermatologist-level classification of skin cancer with deep neural networks",**

A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, et al.,2017.

Skin cancer, the most common human malignancy[1](https://www.nature.com/articles/nature21056#ref-CR1),[2](https://www.nature.com/articles/nature21056#ref-CR2),[3](https://www.nature.com/articles/nature21056#ref-CR3), is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions. Deep convolutional neural networks (CNNs)[4](https://www.nature.com/articles/nature21056#ref-CR4),[5](https://www.nature.com/articles/nature21056#ref-CR5) show potential for general and highly variable tasks across many fine-grained object categories[6](https://www.nature.com/articles/nature21056#ref-CR6),[7](https://www.nature.com/articles/nature21056#ref-CR7),[8](https://www.nature.com/articles/nature21056#ref-CR8),[9](https://www.nature.com/articles/nature21056#ref-CR9),[10](https://www.nature.com/articles/nature21056#ref-CR10),[11](https://www.nature.com/articles/nature21056#ref-CR11). Here we demonstrate classification of skin lesions using a single CNN, trained end-to-end from images directly, using only pixels and disease labels as inputs. We train a CNN using a dataset of 129,450 clinical images—two orders of magnitude larger than previous datasets[12](https://www.nature.com/articles/nature21056#ref-CR12) consisting of 2,032 different diseases. We test its performance against 21 board-certified dermatologists on biopsy-proven clinical images with two critical binary classification use cases: keratinocyte carcinomas versus benign seborrheic keratoses; and malignant melanomas versus benign nevi. The first case represents the identification of the most common cancers, the second represents the identification of the deadliest skin cancer. The CNN achieves performance on par with all tested experts across both tasks, demonstrating an artificial intelligence capable of classifying skin cancer with a level of competence comparable to dermatologists. Outfitted with deep neural networks, mobile devices can potentially extend the reach of dermatologists outside of the clinic. It is projected that 6.3 billion smartphone subscriptions will exist by the year 2021 (ref.) and can therefore potentially provide low-cost universal access to vital diagnostic care.

**"Automated detection and classification of fundus diabetic retinopathy images using synergic deep learning model",**

K. Shankar, A. R. W. Sait, D. Gupta, S. K. Lakshmanaprabu, A. Khanna and H. M. Pandey, 2020.

In recent days, the incidence of Diabetic Retinopathy (DR)has become high, affecting the eyes because of drastic increase in the glucose level in blood. Globally, almost half of the people under the age of 70 gets severely affected by diabetes. In the absence of earlier recognition and proper medication, the DR patients tend to lose their vision. When the warning signs are tracked down, the severity level of the disease has to be validated so to take decisions regarding appropriate treatment further. The current research paper focuses on the concept of classification of DR fundus images on the basis of severity level using a [deep learning model](https://www.sciencedirect.com/topics/computer-science/deep-learning-model). This paper proposes a deep learning-based automated detection and [classification model](https://www.sciencedirect.com/topics/computer-science/classification-models) for fundus DR images. The proposed method involves various processes namely preprocessing, segmentation and classification. The methods begins with preprocessing stage in which unnecessary noise that exists in the edges is removed. Next, histogram-based segmentation takes place to extract the useful regions from the image. Then, Synergic Deep Learning (SDL) model was applied to classify the DR fundus images to various severity levels. The justification for the presented SDL model was carried out on Messi or DR dataset. The experimentation results indicated that the presented SDL model offers better classification over the existing models.

**"Multi-retinal disease classification by reduced deep learning features",**

R. Arunkumar and P. Karthigaikumar, 2017.

This paper presents the retina-based disease diagnosis through deep learning-based feature extraction method. This process helps in developing automated screening system, which is capable of diagnosing retina for diseases such as age-related molecular degeneration, diabetic retinopathy, macular bunker, retinoblastoma, retinal detachment, and retinitis pigmentosa. Some of these diseases share a common characteristic, which makes the classification difficult. In order to overcome the above-mentioned problem, deep learning feature extraction and a multi-class SVM classifier are used. The main contribution of this work is the reducing the dimension of the features required to classify the retinal disease, which enhances the process of reducing the system requirement as well as good performance.

**"Modified Alexnet architecture for classification of diabetic retinopathy images",**

T. Shanthi and R. S. Sabeenian, 2019.

[Diabetic retinopathy](https://www.sciencedirect.com/topics/pharmacology-toxicology-and-pharmaceutical-science/diabetic-retinopathy) (DR) is an illness occurring in the eye due to increase in [blood glucose](https://www.sciencedirect.com/topics/engineering/blood-glucose) level. Among people in the age group of 70, 50% of deaths are attributed to diabetes. Early identification and appropriate treatment can reduce the loss of sight in many DR patients. Once the symptoms of DR are recognized, the severity of the disease should be evaluated for administering the right medication. This paper focuses on the classification of DR fundus images according to the severity of the disease using [convolutional neural network](https://www.sciencedirect.com/topics/computer-science/convolutional-neural-network) with the application of suitable Pooling, SoftMax and Rectified Linear Activation Unit (ReLU) layers to obtain a high level of accuracy. The performance of the proposed algorithm has been validated using Messi or database. In the case of healthy images, images of stage1, stage 2 and stage 3 of diabetic retinopathy, [classification accuracies](https://www.sciencedirect.com/topics/engineering/classification-accuracy) of 96.6% and 96.2%, 95.6% and 96.6% have been achieved.

**"Quantitative classification of eyes with and without intermediate age-related macular degeneration using optical coherence tomography",**

S. Farsiu, S. J. Chiu, R. V. O’Connell, F. A. Folgar, E. Yuan, J. A. Izatt, et al., 2014.

We semiautomatically delineated the retinal pigment epithelium (RPE) and RPE drusen complex (RPEDC, the axial distance from the apex of the drusen and RPE layer to Bruch's membrane) and total retina (TR, the axial distance between the inner limiting and Bruch's membranes) boundaries. We registered and averaged the thickness maps from control subjects to generate a map of "normal" non-AMD thickness. We considered RPEDC thicknesses larger or smaller than 3 standard deviations from the mean as abnormal, indicating drusen or geographic atrophy (GA), respectively. We measured TR volumes, RPEDC volumes, and abnormal RPEDC thickening and thinning volumes for each subject. By using different combinations of these 4 disease indicators, we designed 5 automated classifiers for the presence of AMD on the basis of the generalized linear model regression framework. We trained and evaluated the performance of these classifiers using the leave-one-out method.

**"Drusen associated with aging and age-related macular degeneration contain proteins common to extracellular deposits associated with atherosclerosis elastosis amyloidosis and dense deposit disease",**

R. F. Mullins, S. R. Russell, D. H. Anderson and G. S. Hageman, 2000.

Age-related macular degeneration (AMD), a blinding disorder that compromises central vision, is characterized by the accumulation of extracellular deposits, termed drusen, between the retinal pigmented epithelium and the choroid. Recent studies in this laboratory revealed that vitronectin is a major component of drusen. Because vitronectin is also a constituent of abnormal deposits associated with a variety of diseases, drusen from human donor eyes were examined for compositional similarities with other extracellular disease deposits. Thirty-four antibodies to 29 different proteins or protein complexes were tested for immunoreactivity with hard and soft drusen phenotypes. These analyses provide a partial profile of the molecular composition of drusen. Serum amyloid P component, apolipoprotein E, immunoglobulin light chains, Factor X, and complement proteins (C5 and C5b-9 complex) were identified in all drusen phenotypes. Transcripts encoding some of these molecules were also found to be synthesized by the retina, retinal pigmented epithelium, and/or choroid. The compositional similarity between drusen and other disease deposits may be significant in view of the recently established correlation between AMD and atherosclerosis. This study suggests that similar pathways may be involved in the etiologist of AMD and other age-related diseases.—Mullins, R. F., Russell, S. R., Anderson, D. H., Hageman, G. S. Drusen associated with aging and age-related macular degeneration contain proteins common to extracellular deposits associated with atherosclerosis, elastosis, amyloidosis, and dense deposit disease. *FASEB J.* 14, 835–846 (2000)

**"Progress on retinal image analysis for age related macular degeneration",**

Y. Kanagasingam, A. Bhuiyan, M. D. Abràmoff, R. T. Smith, L. Goldschmidt and T. Y. Wong, 2014.

Age-related macular degeneration (AMD) is the leading cause of [vision loss](https://www.sciencedirect.com/topics/nursing-and-health-professions/visual-impairment) in those over the age of 50 years in the [developed countries](https://www.sciencedirect.com/topics/engineering/developed-country). The number is expected to increase by ∼1.5 fold over the next ten years due to an increase in aging population. One of the main measures of AMD severity is the analysis of [drusen](https://www.sciencedirect.com/topics/pharmacology-toxicology-and-pharmaceutical-science/drusen), pigmentary abnormalities, [geographic atrophy](https://www.sciencedirect.com/topics/medicine-and-dentistry/geographic-atrophy) (GA) and [choroidal neovascularization](https://www.sciencedirect.com/topics/nursing-and-health-professions/subretinal-neovascularization) (CNV) from imaging based on colour fundus [photograph](https://www.sciencedirect.com/topics/medicine-and-dentistry/photograph), optical coherence [tomography](https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/tomography) (OCT) and other [imaging modalities](https://www.sciencedirect.com/topics/engineering/imaging-modality). Each of these imaging modalities has strengths and [weaknesses](https://www.sciencedirect.com/topics/pharmacology-toxicology-and-pharmaceutical-science/weakness) for extracting individual AMD pathology and different [imaging techniques](https://www.sciencedirect.com/topics/medicine-and-dentistry/imaging-technique) are used in combination for capturing and/or quantification of different pathologies. Current dry AMD treatments cannot cure or reverse [vision loss](https://www.sciencedirect.com/topics/medicine-and-dentistry/visual-impairment). However, the Age-Related [Eye Disease](https://www.sciencedirect.com/topics/neuroscience/ophthalmic-disorder) Study (AREDS) showed that specific anti-oxidant [vitamin supplementation](https://www.sciencedirect.com/topics/medicine-and-dentistry/vitamin-supplementation) reduces the risk of progression from intermediate stages (defined as the presence of either many medium-sized [drusen](https://www.sciencedirect.com/topics/medicine-and-dentistry/drusen) or one or more large drusen) to late AMD which allows for preventative strategies in properly identified patients. Thus identification of people with early stage AMD is important to design and implement preventative strategies for late AMD, and determine their cost-effectiveness. A mass screening facility with [teleophthalmology](https://www.sciencedirect.com/topics/nursing-and-health-professions/telemedicine-in-ophthalmology) or [telemedicine](https://www.sciencedirect.com/topics/nursing-and-health-professions/telemedicine) in combination with computer-aided analysis for large rural-based communities may identify more individuals suitable for early stage AMD prevention. In this review, we discuss different imaging modalities that are currently being considered or used for screening AMD. In addition, we look into various automated and semi-automated computer-aided grading systems and related [retinal image](https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/retina-image) analysis techniques for drusen, [geographic atrophy](https://www.sciencedirect.com/topics/pharmacology-toxicology-and-pharmaceutical-science/geographic-atrophy) and [choroidal neovascularization](https://www.sciencedirect.com/topics/neuroscience/choroidal-neovascularization) detection and/or quantification for measurement of AMD severity using these imaging modalities. We also review the existing [telemedicine](https://www.sciencedirect.com/topics/medicine-and-dentistry/telemedicine) studies which include diagnosis and management of AMD, and how automated disease grading could benefit telemedicine. As there is no treatment for dry AMD and only [early intervention](https://www.sciencedirect.com/topics/nursing-and-health-professions/early-intervention) can prevent the late AMD, we emphasize mass screening through a telemedicine platform to enable early detection of AMD. We also provide a comparative study between the imaging modalities and identify potential study areas for further improvement and future research direction in automated AMD grading and screening.

**"Identifying medical diagnoses and treatable diseases by image-based deep learning",**

D. S. Kermany, 2018.

The implementation of clinical-decision support algorithms for medical imaging [faces](https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/face) challenges with reliability and interpretability. Here, we establish a diagnostic tool based on a deep-learning framework for the screening of patients with common treatable blinding [retinal diseases](https://www.sciencedirect.com/topics/neuroscience/retinal-disease). Our framework utilizes, which trains a [neural network](https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/neural-network) with a fraction of the data of conventional approaches. Applying this approach to a dataset of [optical coherence tomography](https://www.sciencedirect.com/topics/materials-science/optical-coherence-tomography) images, we demonstrate performance comparable to that of human experts in classifying age-related macular degeneration and diabetic macular enema. We also provide a more transparent and interpretable diagnosis by highlighting the regions recognized by the [neural network](https://www.sciencedirect.com/topics/neuroscience/neural-network). We further demonstrate the general applicability of our AI system for diagnosis of [paediatric pneumonia](https://www.sciencedirect.com/topics/medicine-and-dentistry/pediatric-pneumonia) using chest X-ray images. This tool may ultimately aid in expediting the diagnosis and referral of these treatable conditions, thereby facilitating earlier treatment, resulting in improved clinical outcomes.

**"A classified and comparative study of edge detection algorithms",**

M. M. M. S. Fathy and M. T. Mahmoudi, 2002.

Since edge detection is in the forefront of image processing for object detection, it is crucial to have a good understanding of edge detection algorithms. This paper introduces a new classification of most important and commonly used edge detection algorithms, namely ISEF, Canny, Marr-Hildreth, Sobel, Kirsch, Lapla1 and Lapla2. Five categories are included in our classification, and then advantages and disadvantages of some available algorithms within this category are discussed. A representative group containing the above seven algorithms are the implemented in C++ and compared subjectively, using 30 images out of 100 images. Two sets of images resulting from the application of those algorithms are then presented. It is shown that under noisy conditions, ISEF, Canny, Marr-Hildreth, Kirsch, Sobel, Lapla2, Lapla1 exhibit better performance, respectively.

**"A novel hybrid machine learning model for auto-classification of retinal diseases",**

C.-H. H. Yang, J.-H. Huang, F. Liu, F.-Y. Chiu, M. Gao, W. Lyu, et al., 2018.

Age-Related Macular Degeneration (AMD) is an asymptomatic retinal disease which may result in loss of vision. There is limited access to high-quality relevant retinal images and poor understanding of the features deﬁning sub-classes of this disease. Motivated by recent advances in machine learning we speciﬁcally explore the potential of generative modelling, using Generative Adversarial Networks (GANs) and style transferring, to facilitate clinical diagnosis and disease understanding by feature extraction. We design an analytic pipeline which ﬁrst generates synthetic retinal images from clinical images; a subsequent veriﬁcation step is applied. In the synthesizing step we merge GANs (DCGANs and WGANs architectures) and style transferring for the image generation, whereas the veriﬁed step controls the accuracy of the generated images. We ﬁnd that the generated images contain suﬃcient pathological details to facilitate ophthalmologists’ task of disease classiﬁcation and in discovery of disease relevant features. In particular, our system predicts the drusen and geographic atrophy sub-classes of AMD. Furthermore, the performance using CFP images for GANs outperforms the classiﬁcation based on using only the original clinical dataset. Our results are evaluated using existing classiﬁer of retinal diseases and class activated maps, supporting the predictive power of the synthetic images and their utility for feature extraction. Our code examples are available online.

**3. SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM :**

Retinal disease are one of the dangerous disease which may cause blindness if not detected and treat early. So to enhance detection accuracy author employed EyeDeep-Net model. The objective of EyeDeep-Net model is to develop a diagnostic framework for the diagnosis of multiple fundus disease at an early stage through a common deep neural network so that people can get treatment on time and take necessary actions to save their eyes from being lost. The main contribution of propose algorithm is to developed robust and effective framework using the proposed EyeDeep-Net model, a deep learning architecture to classify fundus images and diagnose different eye diseases. An open source multi-labelled dataset was transformed into a multiclass dataset and then extracted fundus images were augmented to deal with real-world conditions and processed.

**Disadvantages**

1. It takes more time
2. Less accuracy

**3.2 PROPOSED SYSTEM :**

In propose paper author introducing deep neural network based algorithm to detect retinal diseases. To train network author employing RFMID dataset which contains retinal images of 46 different classes and from this classes author extracting most 4 different classes such as Normal, DR, ODC and MH. Above classes contains highly imbalance data where one class contains a greater number of images and other class contains fewer number of images and such data imbalance may affect accuracy so to enhance accuracy of propose algorithm author employing Data Augmentation techniques which will generate new synthetic images using available images. To generate augmented images author has used left and right rotation of 15 % with sheer rotation as 0.8 and applied horizontal flip.

**EXTENSION CONCEPT:**

In propose EyeDeep-Net model author has used padding as ‘SAME’ which adds additional rows and columns of pixels around the edges of the input data so that the size of the output feature map is the same as the size of the input data. Adding additional rows and columns may hinder algorithm performance so we can apply padding as ‘VALID’ which is used when it is desired to reduce the size of the output feature map in order to reduce the number of parameters in the model and improve its computational efficiency. When features size reduced then algorithm will get more optimized features and accuracy will get enhanced.

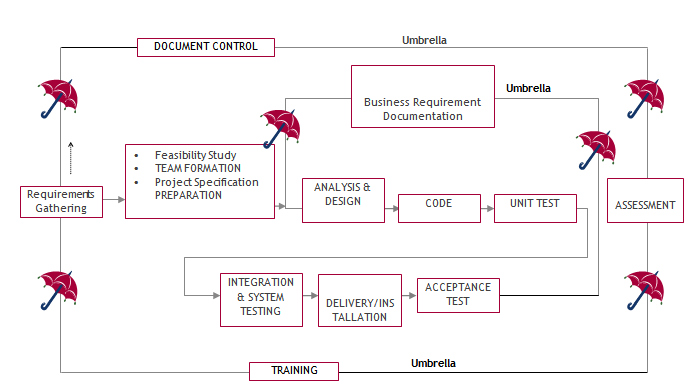
So extension work we have utilized padding same with valid to reduce parameters and to enhance accuracy.

**Advantages**

1. It takes less time
2. High prediction result

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

****

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

1. Requirement Gathering
2. Analysis
3. Designing
4. Coding
5. Testing
6. Maintenance

**Requirements Gathering stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

1. Business requirements describe in business terms what must be delivered or accomplished to provide value.
2. Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
3. Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel I3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500GB

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows 10/above
* Programming Language - Python 3.7.0

**4. SYSTEM DESIGN**

**UML Diagram:**

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* 1. **User Model View**
     1. This view represents the system from the users perspective.
     2. The analysis representation describes a usage scenario from the end-users perspective.
  2. **Structural Model view**
     1. In this model the data and functionality are arrived from inside the system.
     2. This model view models the static structures.

1. **Behavioural Model View**

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

1. **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

1. **Environmental Model View**

In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

**Class Diagram:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake.



**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

****

**Sequence diagram:**

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Collaboration diagram:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**Component Diagram:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

****

**Deployment Diagram:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

****

**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another

activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

**Upload Dataset**

**Preprocess Dataset**

**EyeNet with SGD**

datasetSplit

datasetSplit

datasetSplit

**EyeNet with Adam**

**Train with Adam & Valid Padding**

**Comparision Graph**

**Accuracy & Loss Graph**

**Predict from Test**

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

1. Upload Dataset 2. Dataset Uploaded

3. Preprocess Dataset 4. Dataset Preprocessed

5. EyeNet with SGD 6. EyeNet with SGD Successfully

7. EyeNet with Adam 8. EyeNet with Adam Successfully

9. Train with Adam & Valid Padding 10. Train with Adam & Valid Padding Successfully

11. Comparision Graph 12. Comparision Graph Displayed

13. Accuracy & Loss Graph 14. Accuracy & Loss Graph Displayed

15. Predict from Test 16. Predict from Test Successfully

**5. IMPLEMETATION**

**5.1 PYTHON**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**5.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**5.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language; therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language; therefore the type of the value will be determined at runtime.

**5.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI application Makes the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**5.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**5.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics

**XGBOOST Algorithm:**

**XGBoost** is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

XGBoost can be used in a variety of applications, including Kaggle competitions, recommendation systems, and click-through rate prediction, among others. It is also highly customizable and allows for fine-tuning of various model parameters to optimize performance.

XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting.

**Before understanding the XGBoost, we first need to understand the trees especially the decision tree:**

**Decision Tree:**

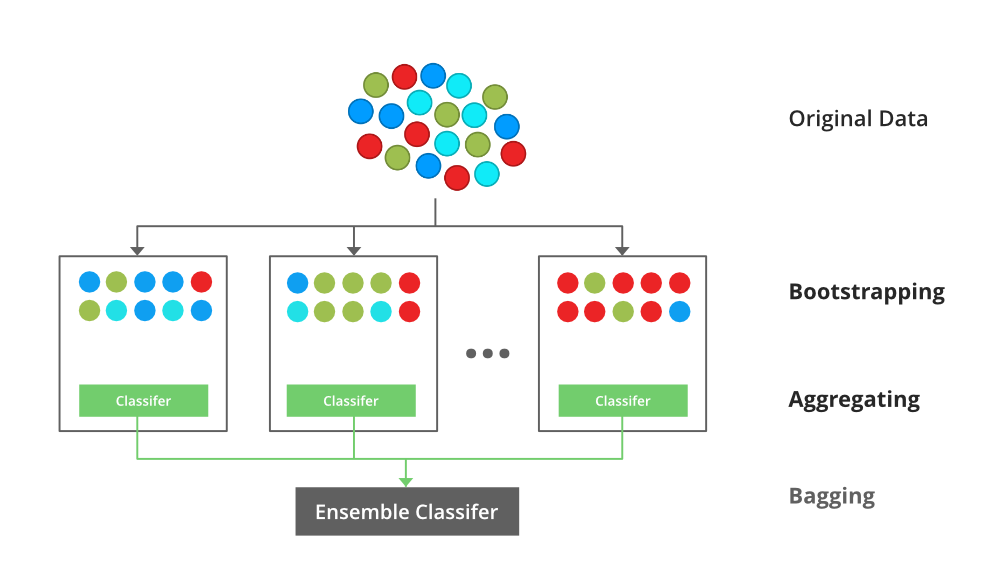
A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

**Bagging**:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.  
Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset, where N is the size of the original training set. The training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.



**Random Forest**:

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn’t depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.rRandom Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

**Boosting**:

Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.



**Gradient Boosting**

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor’s error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

**XGBoost**

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

**Deep neural networks (DNN)**

Deep neural networks (DNN) is a class of [machine learning](https://www.sciencedirect.com/topics/computer-science/machine-learning) algorithms similar to the [artificial neural network](https://www.sciencedirect.com/topics/chemical-engineering/neural-network) and aims to mimic the information processing of the brain. [DNN](https://www.sciencedirect.com/topics/computer-science/deep-neural-network) shave more than one hidden layer (l) situated between the input and output layers (Good fellow et al., 2016). Each layer contains a given number of units (neurons) that apply a certain functional transformation to the input. These types of models can approximate the behavior of any function (universal approximation theorem). The output (y) of a unit (i) in layer (l) is related to the output (x) of the earlier layer (k) with J outputs through a set of weights (w*i,k*), a bias (b) and a non-linear [activation function](https://www.sciencedirect.com/topics/computer-science/activation-function) f.

(3)𝑦𝑖𝑙=𝑓∑𝑗=1𝐽𝑤𝑖,𝑘𝑥𝑘+𝑏𝑖

To fairly compare the various models in this work, only feed-forward layers were used. Despite the considerable interest, DNNs have gained in regression applications, few have dealt with the uncertainty in the prediction. This could be due to the complexity required to perform such an analysis. In this work, we use a probabilistic [machine learning technique](https://www.sciencedirect.com/topics/computer-science/machine-learning-technique) by introducing a probabilistic layer after the dense layers that can learn the distribution over the weights in the network. This layer learns the probability distribution of the outputs (y) related to the inputs (x) through weights (w). This makes it possible to model the loss function as the negative log-likelihood. The DNN was implemented using the TensorFlow frame work along with the TensorFlow distributions to model the probabilistic layer (Dillon et al.,2017). Grid-search was used to optimize the DNN hyperparameter systematically by varying the number of neurons, and dropout rate between [8,16,32,64], and [0.2-0.6] respectively. This resulted in a DNN comprised of: three dense layers (64) separated by three dropout layers (0.5) with a [learning rate](https://www.sciencedirect.com/topics/computer-science/learning-rate) of 1e-4and using the “Adam” optimizer and the “ReLu” activation function. The architecture was determined using an 80:10:10 training, validation, and testing split.

**LSTM**

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well-suited for tasks such as language translation, speech recognition, and time series forecasting. LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.

The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell. This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

**Bidirectional LSTM**

Bidirectional LSTM (Bi LSTM/ BLSTM) is recurrent neural network (RNN) that is able to process sequential data in both forward and backward directions. This allows Bi LSTM to learn longer-range dependencies in sequential data than traditional LSTMs, which can only process sequential data in one direction.

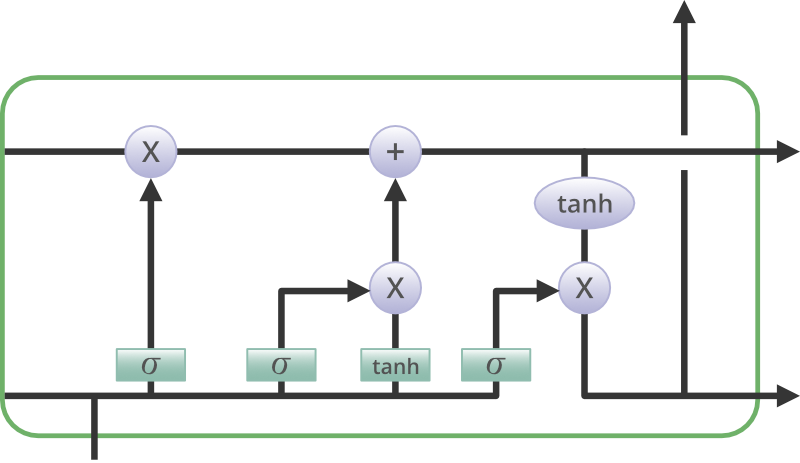
* Bi LSTMs are made up of two LSTM networks, one that processes the input sequence in the forward direction and one that processes the input sequence in the backward direction. The outputs of the two LSTM networks are then combined to produce the final output.

1. Bi LSTM have been shown to achieve state-of-the-art results on a wide variety of tasks, including machine translation, speech recognition, and text summarization.

LSTMs can be stacked to create deep LSTM networks, which can learn even more complex patterns in sequential data. Each LSTM layer captures different levels of abstraction and temporal dependencies in the input data.

**Architecture and Working of LSTM**

LSTM architecture has a chain structure that contains four neural networks and different memory blocks called **cells**.



Information is retained by the cells and the memory manipulations are done by the**gates.** There are three gates –

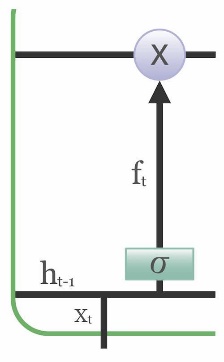
**Forget Gate**

The information that is no longer useful in the cell state is removed with the forget gate. Two inputs *xt* (input at the particular time) and *ht-1* (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use. The equation for the forget gate is:

 where:

* W\_f represents the weight matrix associated with the forget gate.

1. [h\_t-1, x\_t] denotes the concatenation of the current input and the previous hidden state.
2. b\_f is the bias with the forget gate.
3. σ is the sigmoid activation function.



**Input gate**

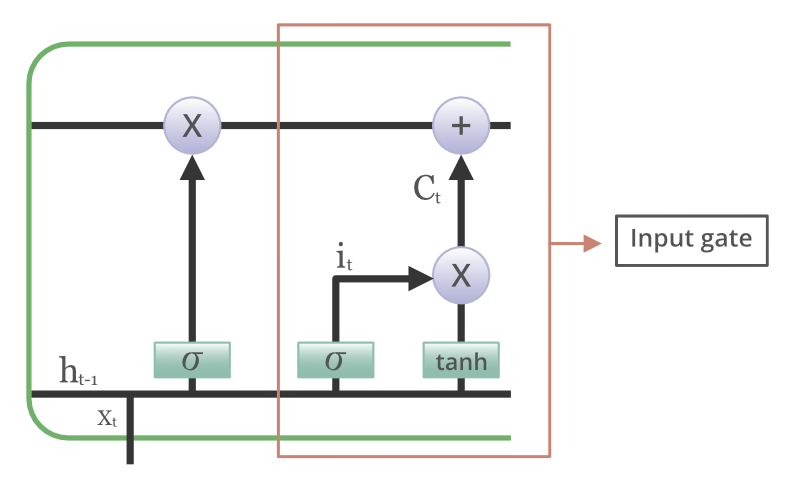
The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs*ht-1* and *xt*. . Then, a vector is created using*tanh*function that gives an output from -1 to +1, which contains all the possible values from ht-1 and *xt*. At last, the values of the vector and the regulated values are multiplied to obtain the useful information. The equation for the input gate is:

We multiply the previous state by ft, disregarding the information we had previously chosen to ignore. Next, we include it∗Ct. This represents the updated candidate values, adjusted for the amount that we chose to update each state value.

where

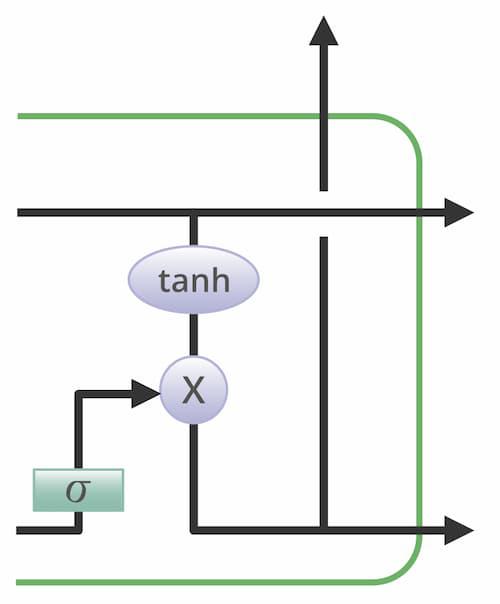
* ⊙ denotes element-wise multiplication

1. tanh is tanh activation function



**Output gate**

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs*ht-1* and *xt*. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell. The equation for the output gate is:



**Convolutional Neural Network (CNN)**

A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use **Recurrent Neural Networks** more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called [feedforward](https://www.geeksforgeeks.org/understanding-multi-layer-feed-forward-networks/), we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called [Backpropagation](https://www.geeksforgeeks.org/backpropagation-in-data-mining/) which basically is used to minimize the loss.

**Convolution Neural Network**

Convolutional Neural Network (CNN) is the extended version of [artificial neural networks (ANN)](https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

**CNN architecture**

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

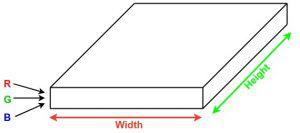


*Simple CNN architecture*

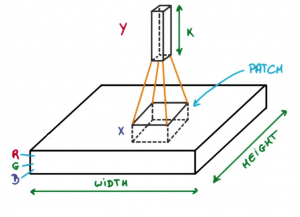
The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

**How Convolutional Layers works**

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).



Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called **Convolution**. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



*Image source: Deep Learning Udacity*

Now let’s talk about a bit of mathematics that is involved in the whole convolution process.

* Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).

1. For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where ‘a’ can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
2. During the forward pass, we slide each filter across the whole input volume step by step where each step is called **stride** (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.
3. As we slide our filters we’ll get a 2-D output for each filter and we’ll stack them together as a result, we’ll get output volume having a depth equal to the number of filters. The network will learn all the filters.

**Layers used to build ConvNets**

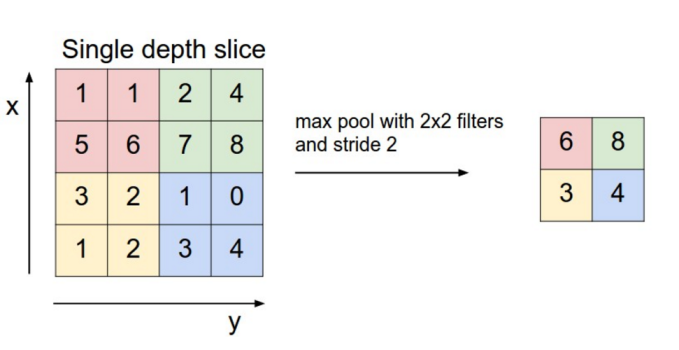
A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

**Types of layers:**

datasets  
Let’s take an example by running a covnets on of image of dimension 32 x 32 x 3.

* **Input Layers:** It’s the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

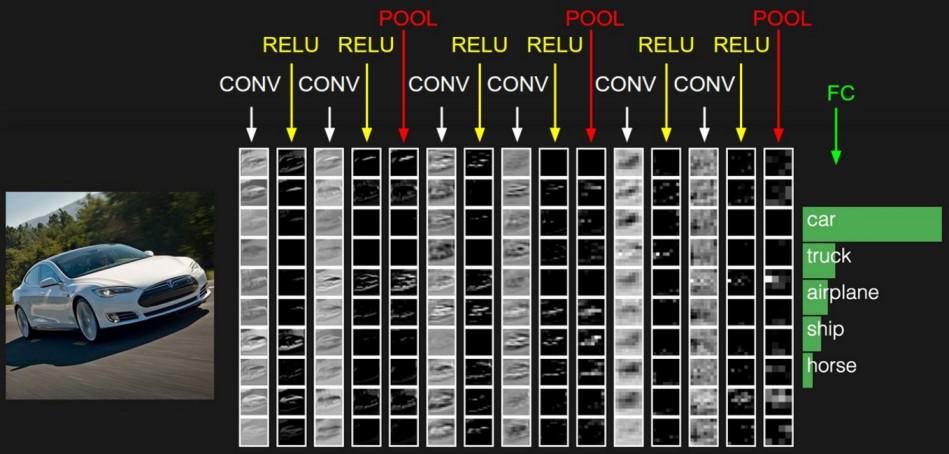
1. **Convolutional Layers:**This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we’ll get an output volume of dimension 32 x 32 x 12.
2. **Activation Layer:**By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are **RELU**: max(0, x),  **Tanh**, **Leaky RELU**, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.
3. **Pooling layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.



*Image source: cs231n.stanford.edu*

* **Flattening:**The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

1. **Fully Connected Layers:**It takes the input from the previous layer and computes the final classification or regression task.



* **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

**5.2 Sample Code:**

**Main.py**

#importing require python packages and classes

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import cv2

import numpy as np

import os

from keras.preprocessing.image import ImageDataGenerator

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from keras.utils.np\_utils import to\_categorical

import pickle

from sklearn.metrics import confusion\_matrix #class to calculate accuracy and other metrics

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

import seaborn as sns

import matplotlib.pyplot as plt

from keras.callbacks import ModelCheckpoint

import keras

from keras import Model, layers

import pandas as pd

from keras.optimizers import SGD #import SGD class

from keras.optimizers import Adam #import Adam class optimizer

from keras.utils.np\_utils import to\_categorical

from keras.models import Sequential

from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, InputLayer, BatchNormalization, Dropout

main = tkinter.Tk()

main.title("EyeDeep-Net")

main.geometry("1300x1200")

#define global variables to calculate and store accuracy and other metrics

precision = []

recall = []

fscore = []

accuracy = []

global filename, dataset,testImages,testLabels

global data,labels,xlabel,ylabel,extension\_gru,extension\_model

global X, Y,data

global X\_train, X\_test, y\_train, y\_test,eyenet\_model

global X\_train, X\_val, y\_train, y\_val

#define global variables

X = []

Y = []

path = "SelectedImages"

labels = []

#define function to load class labels

for root, dirs, directory in os.walk(path):

for j in range(len(directory)):

name = os.path.basename(root)

if name not in labels:

labels.append(name.strip())

def getLabel(name):

index = -1

for i in range(len(labels)):

if labels[i] == name:

index = i

break

return index

print("Retinal Diseases Class Labels found in dataset : "+str(labels))

def UploadDataset():

global filename, dataset, labels, X\_train, Y\_train, text,X,Y

text.delete('1.0', END)

filename = filedialog.askdirectory(initialdir='Dataset')

text.insert(END,filename+" loaded\n\n")

if os.path.exists('model/X.txt.npy'):#if images already processed then load all images

X = np.load('model/X.txt.npy')

Y = np.load('model/Y.txt.npy')

else:#if not processed then read and process each image

X = []

Y = []

for root, dirs, directory in os.walk(path):

for j in range(len(directory)):

name = os.path.basename(root)

if 'Thumbs.db' not in directory[j]:

img = cv2.imread(root+"/"+directory[j])#read image

img = cv2.resize(img, (32, 32))#resize image

X.append(img)#addin images features to training array

label = getLabel(name)

Y.append(label)

X = np.asarray(X)

Y = np.asarray(Y)

np.save('model/X.txt',X)

np.save('model/Y.txt',Y)

text.insert(END,"Dataset Loading Completed"+"\n")

text.insert(END,"Total images found in dataset Before Augmentation : "+str(X.shape[0])+"\n")

#visualizing class labels count found in dataset

names, count = np.unique(Y, return\_counts = True)

height = count

bars = labels

y\_pos = np.arange(len(bars))

plt.figure(figsize = (5, 3))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.xlabel("Dataset Class Label Graph Before Augmentation")

plt.ylabel("Count")

plt.xticks(rotation=90)

plt.show()

def PreprocessDataset():

global filename, dataset,labels,vectorizer

global X, Y

global X\_train, X\_test, y\_train, y\_test

global X\_train, X\_val, y\_train, y\_val

#dataset preprocessing such as shuffling and normalization

text.delete('1.0', END)

#preprocess images like shuffling and normalization

X = X.astype('float32')

X = X/255 #normalized pixel values between 0 and 1

indices = np.arange(X.shape[0])

np.random.shuffle(indices) #shuffle all images

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2) #split dataset into train and test

#now apply augmentation on splitted train images

if os.path.exists('model/aug\_X.txt.npy'):

X = np.load('model/aug\_X.txt.npy')

Y = np.load('model/aug\_Y.txt.npy')

else:

aug = ImageDataGenerator(rotation\_range=15, shear\_range=0.8, horizontal\_flip=True)#apply augmentation to increase images

data = aug.flow(X\_train, y\_train, 1)

X = []

Y = []

for x, y in data:

x = x[0]

y = y[0]

X.append(x)

Y.append(y)

if len(Y) > 30000:

break

X = np.asarray(X)

Y = np.asarray(Y)

np.save('model/aug\_X.txt',X)

np.save('model/aug\_Y.txt',Y)

text.insert(END,"Image Augmentation Completed")

text.insert(END,"Total images found in dataset After Augmentation : "+str(X.shape[0])+"\n")

#split dataset into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2)

text.insert(END,"Training Images Size : "+str(X\_train.shape[0])+"\n")

text.insert(END,"Validation Images Size : "+str(X\_val.shape[0])+"\n")

text.insert(END,"Testing Images Size : "+str(X\_test.shape[0])+"\n")

#plotting graph of augmented images class labels

names, count = np.unique(np.argmax(Y, axis=1), return\_counts=True)

height = count

bars = labels

y\_pos = np.arange(len(bars))

plt.figure(figsize = (5, 3))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.xlabel("Dataset Class Label Graph After Augmentation")

plt.ylabel("Count")

plt.xticks(rotation=90)

plt.show()

#function to calculate various metrics such as accuracy, precision etc

def calculateMetrics(algorithm, predict, testY):

text.delete('1.0', END)

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

text.insert(END,algorithm+' Accuracy : '+str(a)+"\n")

text.insert(END,algorithm+' Precision : '+str(p)+"\n")

text.insert(END,algorithm+' Recall : '+str(r)+"\n")

text.insert(END,algorithm+' FMeasure : '+str(f))

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(5, 3))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.xticks(rotation=90)

plt.show()

def traineyenet():

global filename, dataset

global X, Y

global X\_train, X\_test, y\_train, y\_test

global X\_train, X\_val, y\_train, y\_val,eyenet\_model

#dataset preprocessing such as shuffling and normalization

text.delete('1.0', END)

#train eyenet model using SGD optimizer fixed learning rate

eyenet\_model = keras.models.Sequential([

keras.layers.Conv2D(filters=32, kernel\_size=(11,11), strides=(4,4), activation='relu', input\_shape=(X\_train.shape[1],X\_train.shape[2],X\_train.shape[3])),

keras.layers.BatchNormalization(),

keras.layers.MaxPool2D(pool\_size=(1,1), strides=(2,2)),

keras.layers.Conv2D(filters=16, kernel\_size=(9,9), strides=(1,1), activation='relu', padding="same"),

keras.layers.BatchNormalization(),

keras.layers.MaxPool2D(pool\_size=(1,1), strides=(2,2)),

keras.layers.Conv2D(filters=8, kernel\_size=(7,7), strides=(1,1), activation='relu', padding="same"),

keras.layers.BatchNormalization(),

keras.layers.MaxPool2D(pool\_size=(1,1), strides=(2,2)),

keras.layers.Conv2D(filters=8, kernel\_size=(6,6), strides=(1,1), activation='relu', padding="same"),

keras.layers.BatchNormalization(),

keras.layers.MaxPool2D(pool\_size=(1,1), strides=(2,2)),

keras.layers.Conv2D(filters=8, kernel\_size=(5,5), strides=(1,1), activation='relu', padding="same"),

keras.layers.BatchNormalization(),

keras.layers.MaxPool2D(pool\_size=(1,1), strides=(2,2)),

keras.layers.Conv2D(filters=8, kernel\_size=(3,3), strides=(1,1), activation='relu', padding="same"),

keras.layers.BatchNormalization(),

keras.layers.Conv2D(filters=8, kernel\_size=(3,3), strides=(1,1), activation='relu', padding="same"),

keras.layers.BatchNormalization(),

keras.layers.Conv2D(filters=8, kernel\_size=(3,3), strides=(2,2), activation='relu', padding="same"),

keras.layers.BatchNormalization(),

keras.layers.MaxPool2D(pool\_size=(1,1), strides=(2,2)),

keras.layers.Flatten(),

keras.layers.Dense(64, activation='relu'),

keras.layers.Dropout(0.2),

keras.layers.Dense(64, activation='relu'),

keras.layers.Dropout(0.2),

keras.layers.Dense(y\_train.shape[1], activation='softmax')

])

#compiling, training and loading the model

opt = SGD(lr=0.001)

eyenet\_model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])#compiling the model

if os.path.exists("model/sgd\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/sgd\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = eyenet\_model.fit(X\_train, y\_train, epochs = 40, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/sgd\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

eyenet\_model.load\_weights("model/sgd\_weights.hdf5")

#perfrom prediction on test data

predict = eyenet\_model.predict(X\_val)

predict = np.argmax(predict, axis=1)

y\_val1 = np.argmax(y\_val, axis=1)

acc = accuracy\_score(y\_val1, predict) \* 100

print("EyeNet SGD Validation Accuracy : "+str(acc))

predict = eyenet\_model.predict(X\_test)

predict = np.argmax(predict, axis=1)

y\_test1 = np.argmax(y\_test, axis=1)

calculateMetrics("EyeNet SGD Testing", predict, y\_test1)

def traineyenet1():

global filename, dataset

global X, Y

global X\_train, X\_test, y\_train, y\_test

global X\_train, X\_val, y\_train, y\_val,eyenet\_model

#dataset preprocessing such as shuffling and normalization

text.delete('1.0', END)

#training EyeNet with Adam Optimizer

opt = Adam(lr=0.001)#defining Adam

eyenet\_model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])#compiling the model

#compiling, training and loading the model

if os.path.exists("model/adam\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/adam\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = eyenet\_model.fit(X\_train, y\_train, epochs = 40, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/adam\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

eyenet\_model.load\_weights("model/adam\_weights.hdf5")

#perfrom prediction on test data

predict = eyenet\_model.predict(X\_val)

predict = np.argmax(predict, axis=1)

y\_val1 = np.argmax(y\_val, axis=1)

acc = accuracy\_score(y\_val1, predict) \* 100

print("EyeNet Adam Validation Accuracy : "+str(acc))

predict = eyenet\_model.predict(X\_test)

predict = np.argmax(predict, axis=1)

y\_test1 = np.argmax(y\_test, axis=1)

calculateMetrics("EyeNet Adam Testing", predict, y\_test1)

def traineyenet2():

global filename, dataset,extension\_model

global X, Y

global X\_train, X\_test, y\_train, y\_test

global X\_train, X\_val, y\_train, y\_val

#dataset preprocessing such as shuffling and normalization

text.delete('1.0', END)

#train extension model with padding as 'same and valid'

extension\_model = Sequential()

extension\_model.add(InputLayer(input\_shape=(X\_train.shape[1],X\_train.shape[2],X\_train.shape[3])))

extension\_model.add(Conv2D(25, (5, 5), activation='relu', strides=(1, 1), padding='same'))

extension\_model.add(MaxPool2D(pool\_size=(2, 2), padding='same'))

extension\_model.add(Conv2D(50, (5, 5), activation='relu', strides=(2, 2), padding='same'))

extension\_model.add(MaxPool2D(pool\_size=(2, 2), padding='same'))

extension\_model.add(BatchNormalization())

extension\_model.add(Conv2D(70, (3, 3), activation='relu', strides=(2, 2), padding='same'))

extension\_model.add(MaxPool2D(pool\_size=(2, 2), padding='valid'))

extension\_model.add(BatchNormalization())

extension\_model.add(Flatten())

extension\_model.add(Dense(units=100, activation='relu'))

extension\_model.add(Dense(units=100, activation='relu'))

extension\_model.add(Dropout(0.25))

extension\_model.add(Dense(units=y\_train.shape[1], activation='softmax'))

extension\_model.compile(loss='categorical\_crossentropy', optimizer="adam", metrics=['accuracy'])

if os.path.exists("model/extension\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/extension\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = extension\_model.fit(X\_train, y\_train, epochs = 40, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/extension\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

extension\_model.load\_weights("model/extension\_weights.hdf5")

#perfrom prediction on test data

predict = extension\_model.predict(X\_val)

predict = np.argmax(predict, axis=1)

y\_val1 = np.argmax(y\_val, axis=1)

acc = accuracy\_score(y\_val1, predict) \* 100

print("Extension with Adam & Valid Padding Validation Accuracy : "+str(acc))

predict = extension\_model.predict(X\_test)

predict = np.argmax(predict, axis=1)

y\_test1 = np.argmax(y\_test, axis=1)

calculateMetrics("Extension with Adam & Valid Padding Testing", predict, y\_test1)

def AccuracyGraph():

global filename, dataset,labels

global X, Y

global X\_train, X\_test, y\_train, y\_test

global X\_train1,X\_test1,y\_train1,y\_test1,rf\_cls

#dataset preprocessing such as shuffling and normalization

text.delete('1.0', END)

sgd\_acc, sgd\_loss = values("model/sgd\_history.pckl", "accuracy", "loss")

adam\_acc, adam\_loss = values("model/adam\_history.pckl", "accuracy", "loss")

extension\_acc, extension\_loss = values("model/extension\_history.pckl", "accuracy", "loss")

plt.figure(figsize=(6,4))

plt.grid(True)

plt.xlabel('EPOCH')

plt.ylabel('Accuracy')

plt.plot(sgd\_acc, 'ro-', color = 'green')

plt.plot(adam\_acc, 'ro-', color = 'blue')

plt.plot(extension\_acc, 'ro-', color = 'black')

plt.legend(['EyeDeepNet SGD', 'EyeDeepNet Adam', 'Extension with Valid & Same padding'], loc='lower right')

plt.title('All Algorithm Training Accuracy Graph')

plt.show()

def graph():

text.delete('1.0', END)

#display all algorithm performnace

algorithms = ['EyeNet with SGD', 'EyeNet with Adam', 'Extension with Adam & Valid']

data = []

for i in range(len(accuracy)):

data.append([algorithms[i], accuracy[i], precision[i], recall[i], fscore[i]])

data = pd.DataFrame(data, columns=['Algorithm Name', 'Accuracy', 'Precision', 'Recall', 'FSCORE'])

text.insert(END,str(data)+"\n")

df = pd.DataFrame([['EyeNet with SGD','Accuracy',accuracy[0]],['EyeNet with SGD','Precision',precision[0]],['EyeNet with SGD','Recall',recall[0]],['EyeNet with SGD','FSCORE',fscore[0]],

['EyeNet with Adam','Accuracy',accuracy[1]],['EyeNet with Adam','Precision',precision[1]],['EyeNet with Adam','Recall',recall[1]],['EyeNet with Adam','FSCORE',fscore[1]],

['Extension with Adam & Valid','Accuracy',accuracy[2]],['Extension with Adam & Valid','Precision',precision[2]],['Extension with Adam & Valid','Recall',recall[2]],['Extension with Adam & Valid','FSCORE',fscore[2]],

],columns=['Parameters','Algorithms','Value'])

df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar', figsize=(8, 2))

plt.title("All Algorithms Performance Graph")

plt.show()

def values(filename, acc, loss):

f = open(filename, 'rb')

train\_values = pickle.load(f)

f.close()

accuracy\_value = train\_values[acc][0:20]

loss\_value = train\_values[loss][0:20]

return accuracy\_value, loss\_value

def predict():

global labels

global filename, dataset, X\_train, X\_test, y\_train, y\_test, X, Y, scaler, pca

global accuracy, precision, recall, fscore, values, text,vectorizer

global extension\_model

text.delete('1.0', END)

image\_path = filedialog.askopenfilename(initialdir='.')

image = cv2.imread(image\_path)#read test image

img = cv2.resize(image, (32,32))#resize image

im2arr = np.array(img)

im2arr = im2arr.reshape(1,32,32,3)#convert image as 4 dimension

img = np.asarray(im2arr)

img = img.astype('float32')#convert image features as float

img = img/255 #normalized image

predict = extension\_model.predict(img)#perform prediction on test image

predict = np.argmax(predict)

img = cv2.imread(image\_path)

img = cv2.resize(img, (400,300))#display image with predicted output

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

cv2.putText(img, 'Predicted As : '+labels[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 0, 255), 2)

plt.figure(figsize=(4,3))

plt.imshow(img)

plt.show()

font = ('times', 14, 'bold')

title = Label(main, text='EyeDeep-Net: a multi-class diagnosis of retinal diseases using deep neural network')

title.config(bg='dim gray', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Dataset", bg='dim gray', fg='white',command=UploadDataset)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

font1 = ('times', 13, 'bold')

preprocessButton = Button(main, text="Preprocess Dataset", bg='dim gray', fg='white',command=PreprocessDataset)

preprocessButton.place(x=50,y=150)

preprocessButton.config(font=font1)

font1 = ('times', 13, 'bold')

RunCNNButton = Button(main, text="EyeNet with SGD", bg='dim gray', fg='white',command=traineyenet)

RunCNNButton.place(x=50,y=200)

RunCNNButton.config(font=font1)

font1 = ('times', 13, 'bold')

RunCNNButton = Button(main, text="EyeNet with Adam", bg='dim gray', fg='white',command=traineyenet1)

RunCNNButton.place(x=50,y=250)

RunCNNButton.config(font=font1)

font1 = ('times', 13, 'bold')

RunExtensionButton = Button(main, text="Train with Adam & Valid Padding", bg='dim gray', fg='white',command=traineyenet2)

RunExtensionButton.place(x=50,y=300)

RunExtensionButton.config(font=font1)

font1 = ('times', 13, 'bold')

PButton = Button(main, text="Comparision Graph", bg='dim gray', fg='white',command=graph)

PButton.place(x=50,y=350)

PButton.config(font=font1)

font1 = ('times', 13, 'bold')

RunGraphButton = Button(main, text="Accuracy & Loss Graph", bg='dim gray', fg='white',command=AccuracyGraph)

RunGraphButton.place(x=50,y=400)

RunGraphButton.config(font=font1)

font1 = ('times', 13, 'bold')

PdButton = Button(main, text="Predict from Test", bg='dim gray', fg='white',command=predict)

PdButton.place(x=50,y=450)

PdButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=118)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=330,y=100)

text.config(font=font1)

main.config(bg='Maroon1')

main.mainloop()

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

**Implementation**

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

**Testing**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

**System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

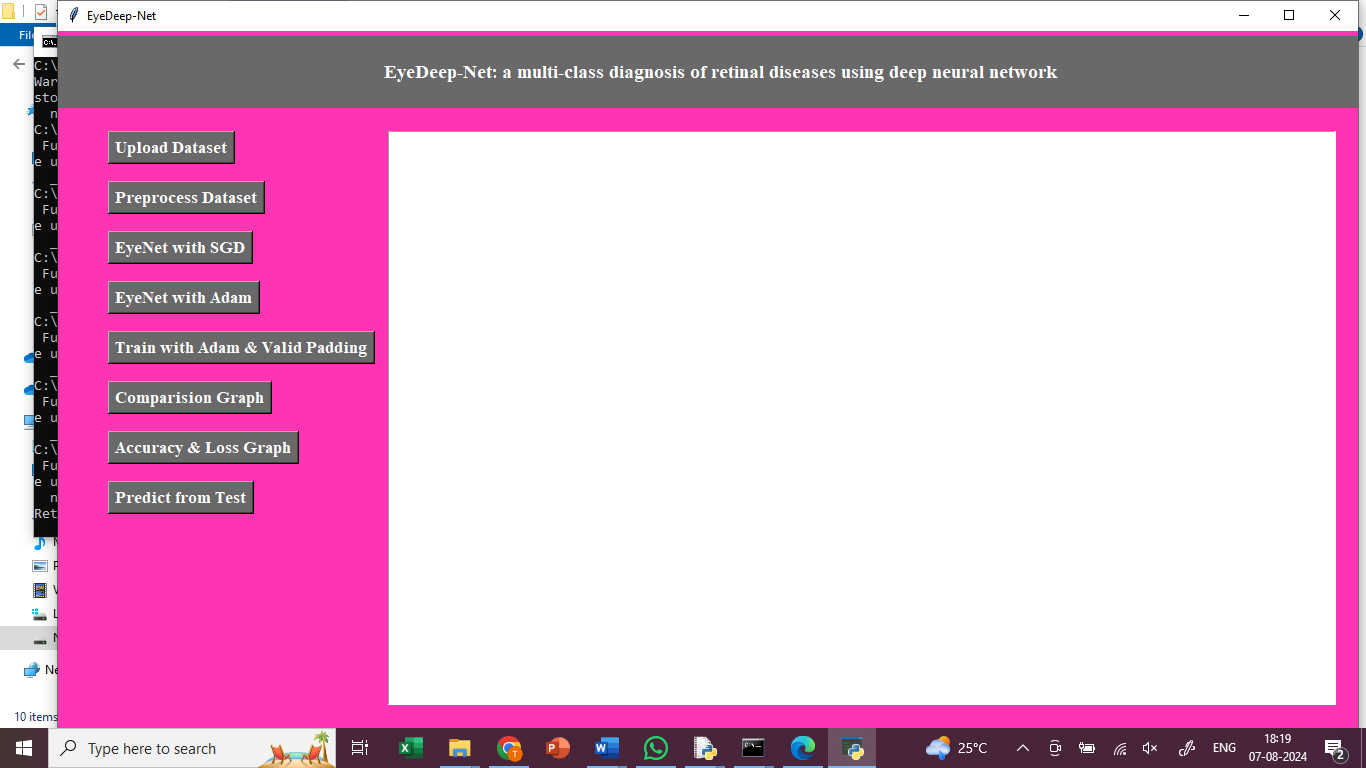
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation

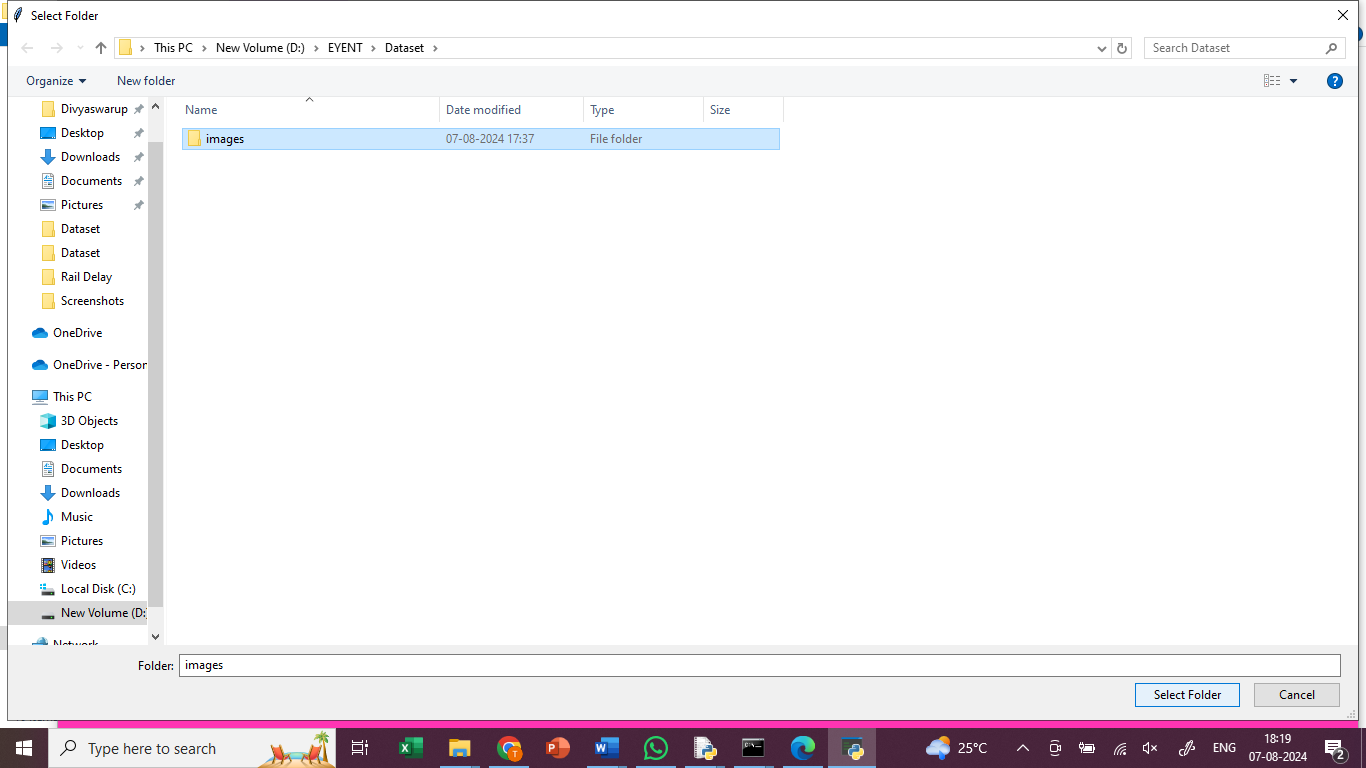
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload Dataset | Test whether Dataset is uploaded or not into the system | If the Dataset may not uploaded | We cannot do further operations | Dataset uploaded we will do further operations | High | High |
| 02 | Preprocess Dataset | Test whether the Pre-process Dataset Successfully or not | If the Pre-process Dataset may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 03 | EyeNet with SGD | Test whether EyeNet with SGD Run Successfully or not | If the  EyeNet with SGDmay not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 04 | EyeNet with Adam | Test whether EyeNet with Adam Run Successfully or not | If the EyeNet with Adam may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 05 | Train with Adam & Valid Padding | Test whether Train with Adam & Valid Padding Run Successfully or not | If the Train with Adam & Valid Padding may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 06 | Comparision Graph | Test whether Comparision Graph Display Successfully or not | If the Comparision Graph may not display Successfully | We cannot do further operations | we will do further  operations | High | High |
| 07 | Accuracy & Loss Graph | Test whether Accuracy & Loss Graph display Successfully or not | If the Accuracy & Loss Graph may not display Successfully | We cannot do further operations | we will do further  operations | High | High |
| 08 | Predict from Test | Test whether Prediction gives Successfully or not | If the Prediction may not give Successfully | We cannot do further operations | we will do further  operations | High | High |

**7.SCREENSHOTS**:

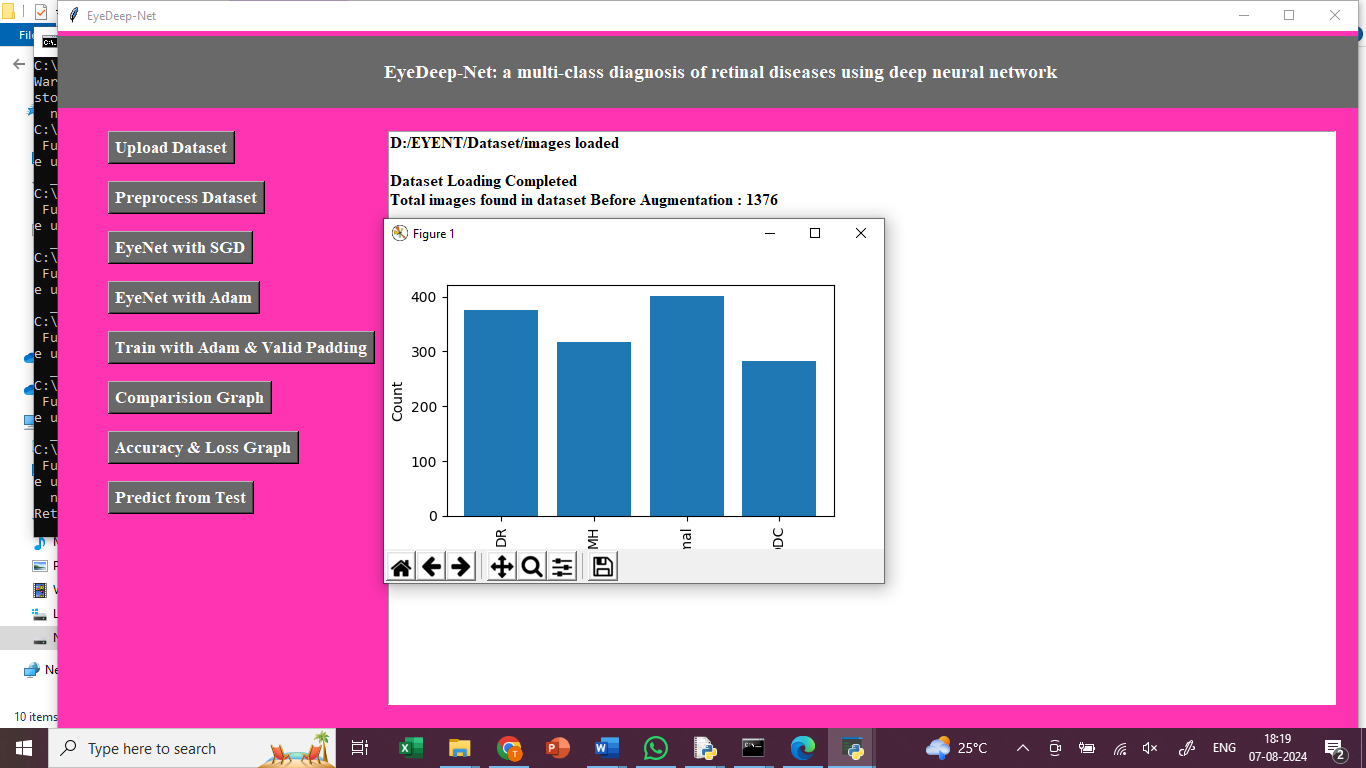
To run project double click on ‘run.bat’ file to get below screen



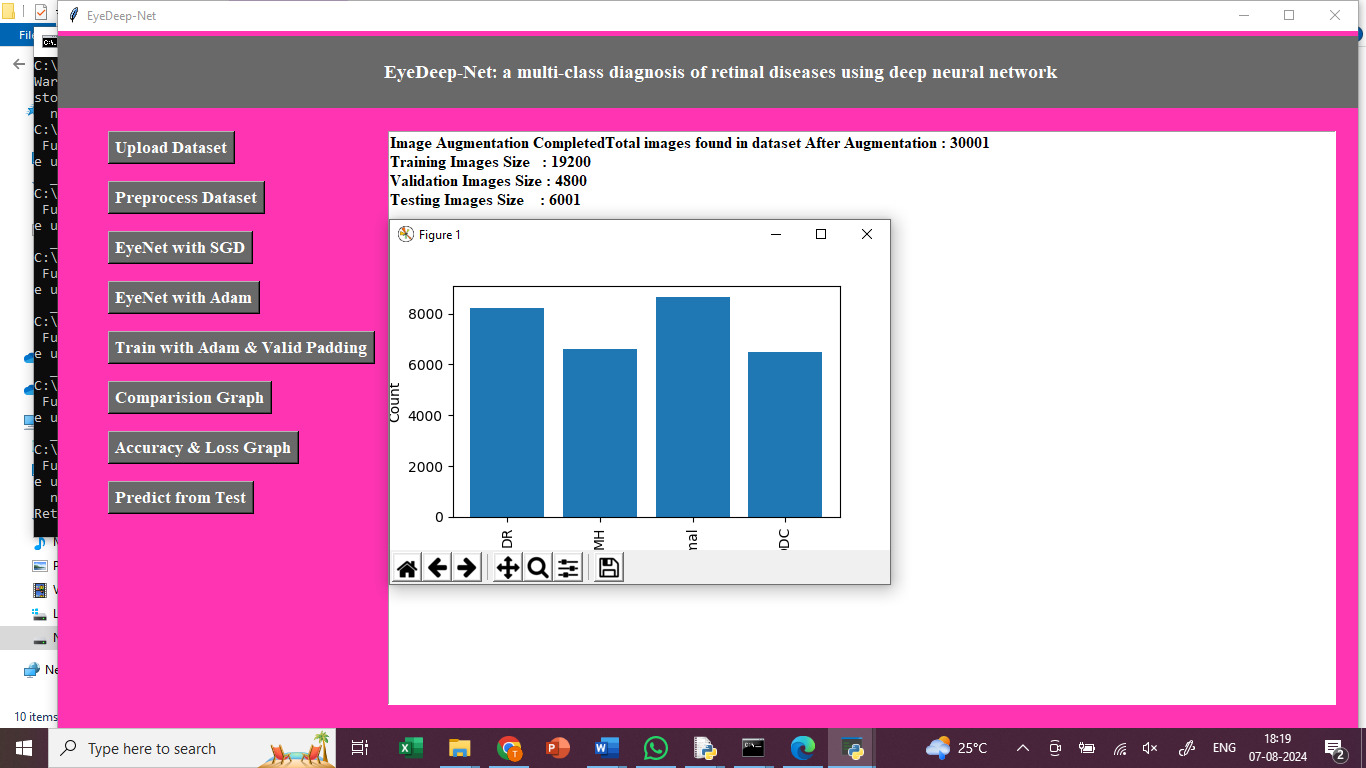
In above screen, click on upload Dataset button



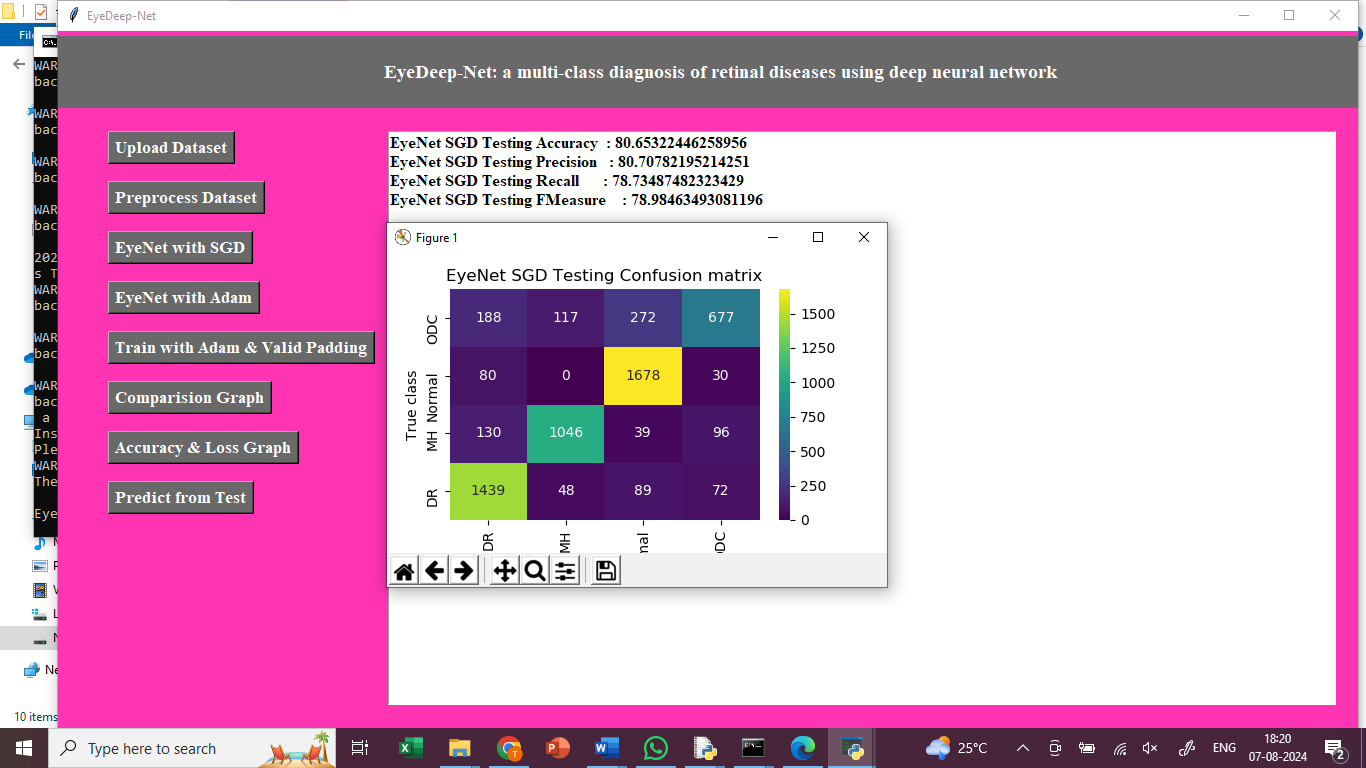
In above screen, upload images



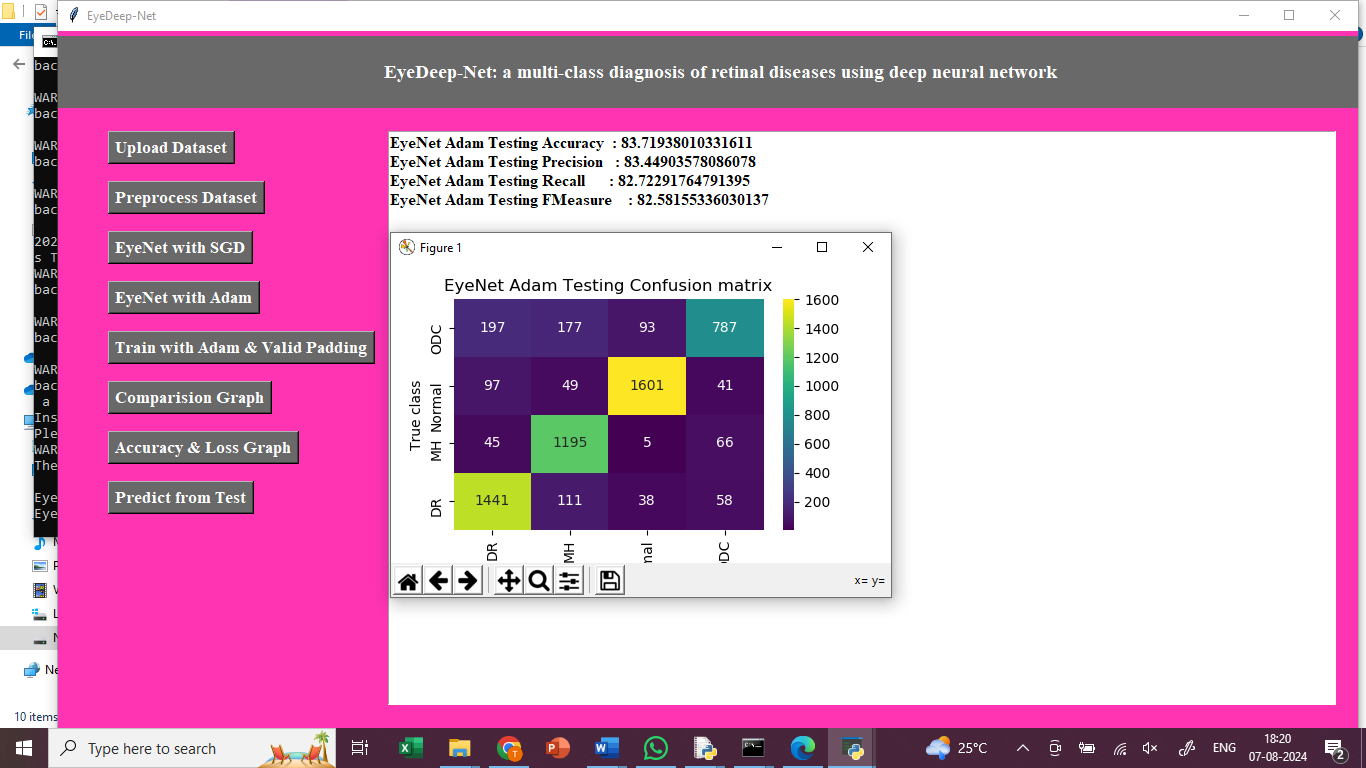
In above screen, click on Preprocess Dataset button



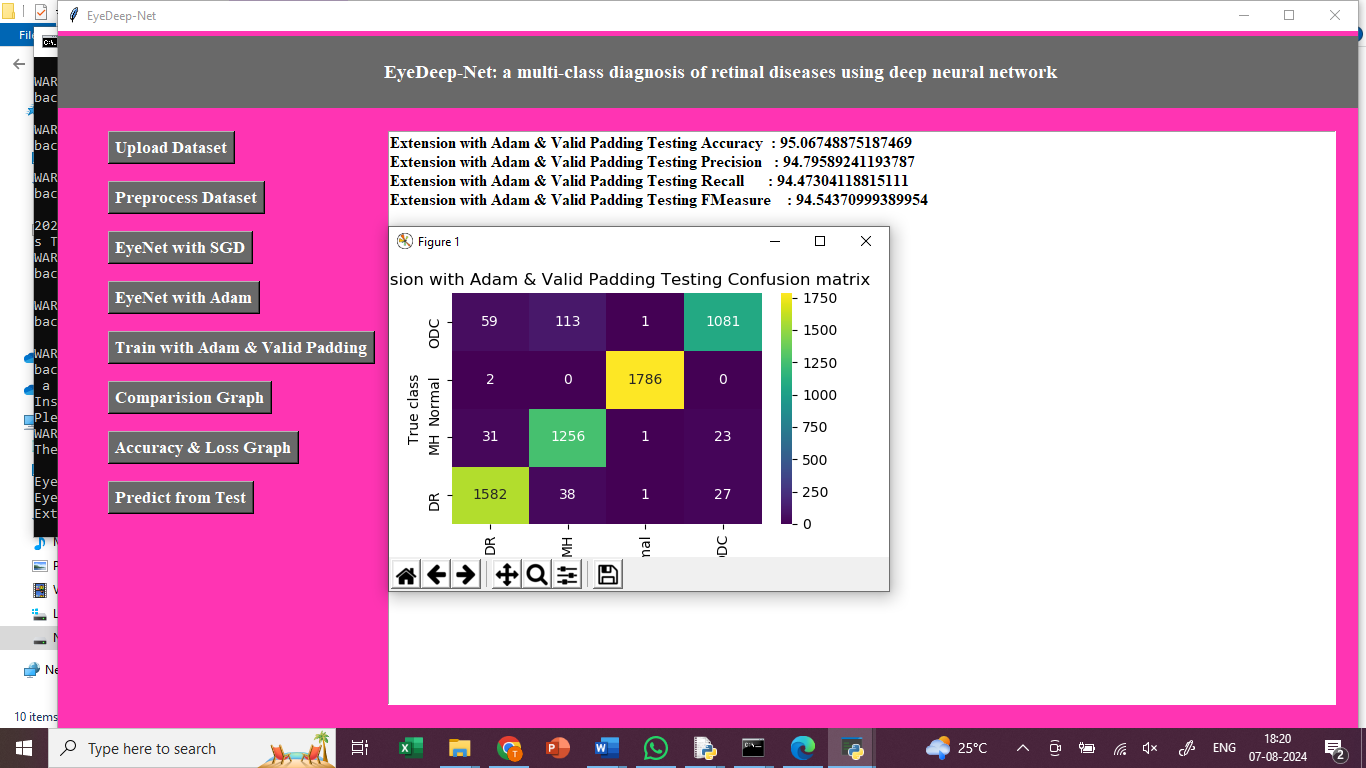
In above screen, for splitting the dataset, 80% for training and 20% for testing.



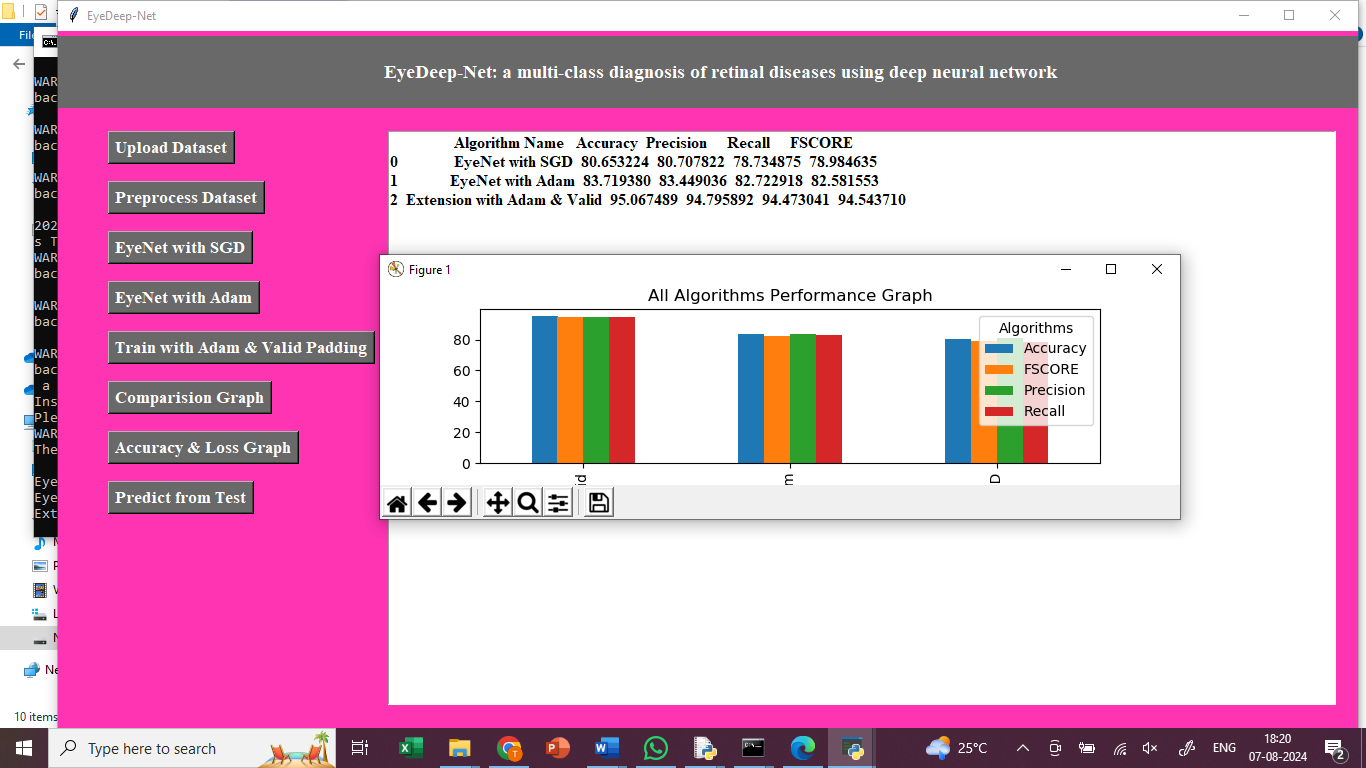
In above screen, click on EyeNet with SGD button, to test EyeNet SGD and got 80% accuracy. In confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels where blue boxes represents incorrect prediction count which are very few and yellow and light green boxes represents correct prediction count.



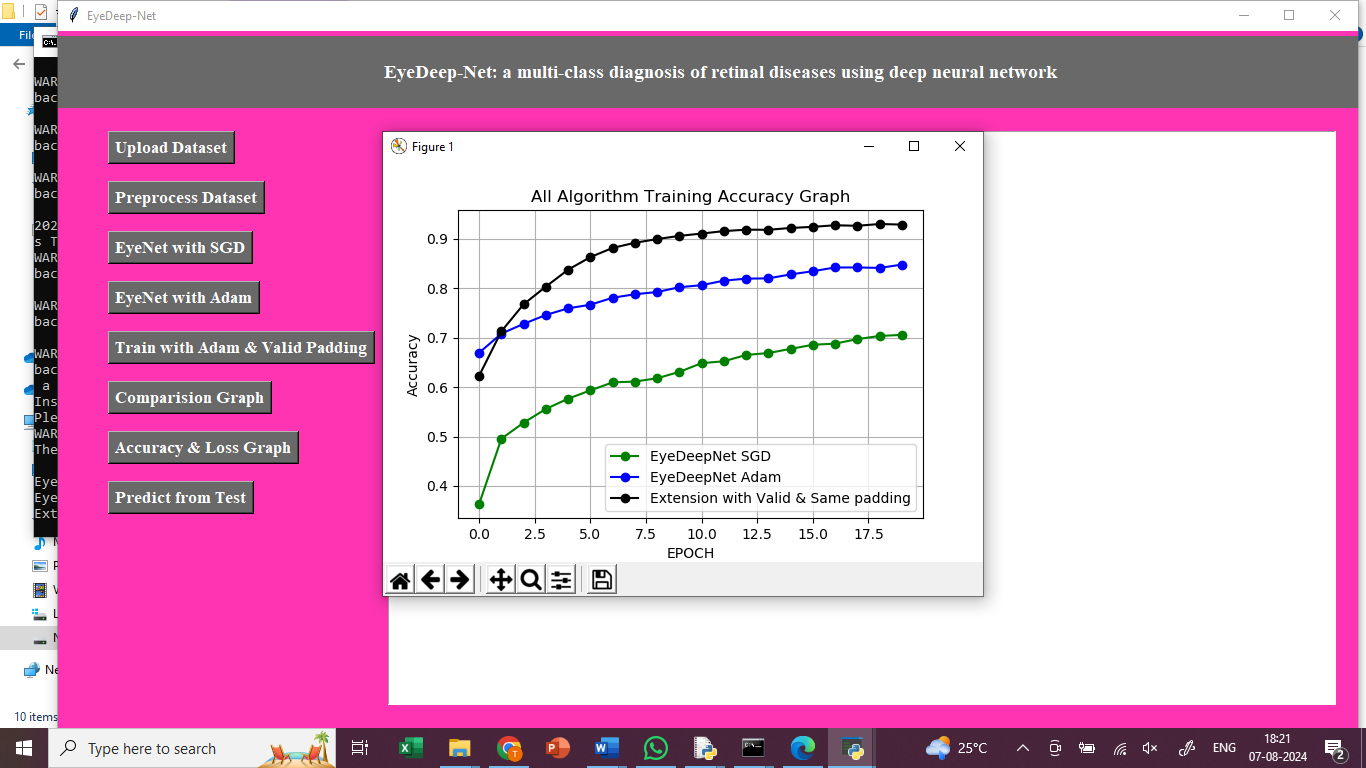
In above screen, click on EyeNet with Adam button, to test EyeNet Adam and got 83% accuracy. In confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels where blue boxes represents incorrect prediction count which are very few and yellow and light green boxes represents correct prediction count.



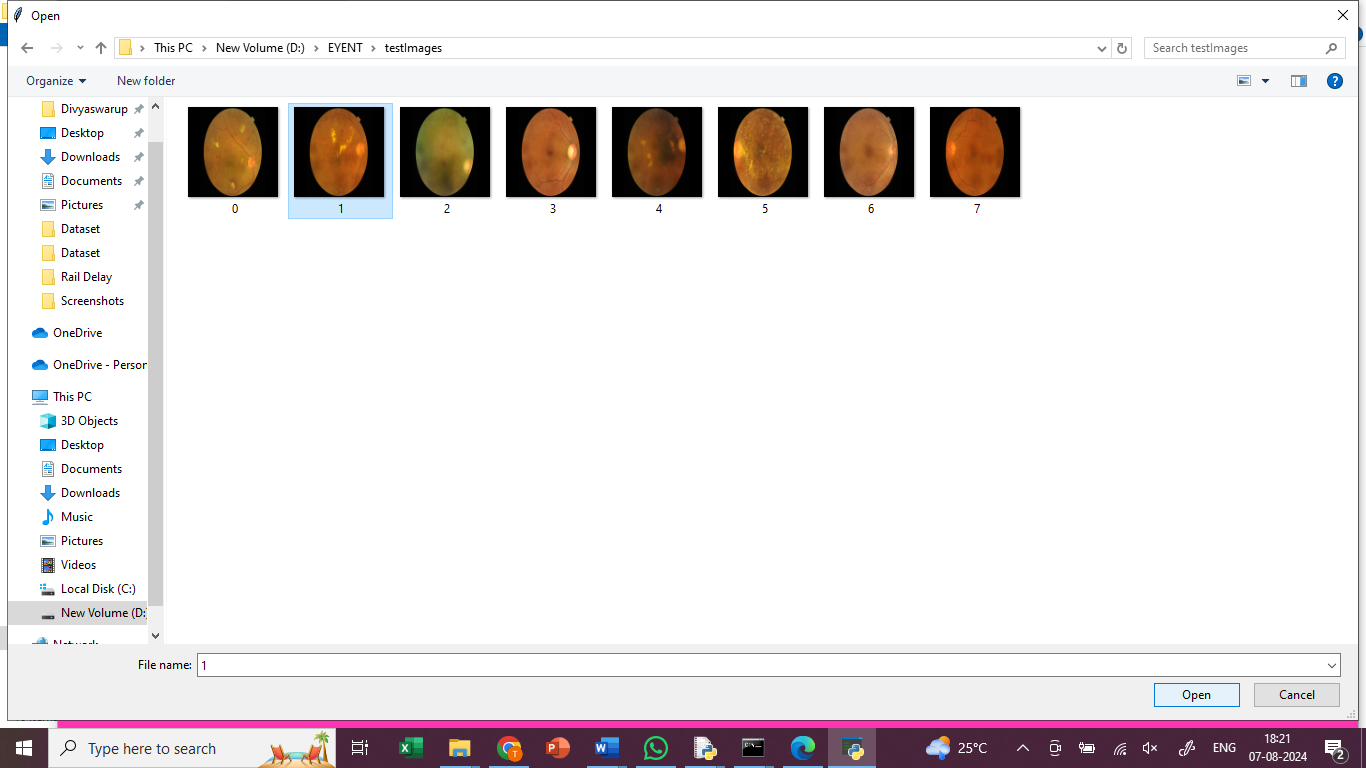
In above screen, click on Train with Adam & Valid Padding button, to test Extension with Adam & Valid Padding and got 95% accuracy. In confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels where blue boxes represents incorrect prediction count which are very few and yellow and light green boxes represents correct prediction count.



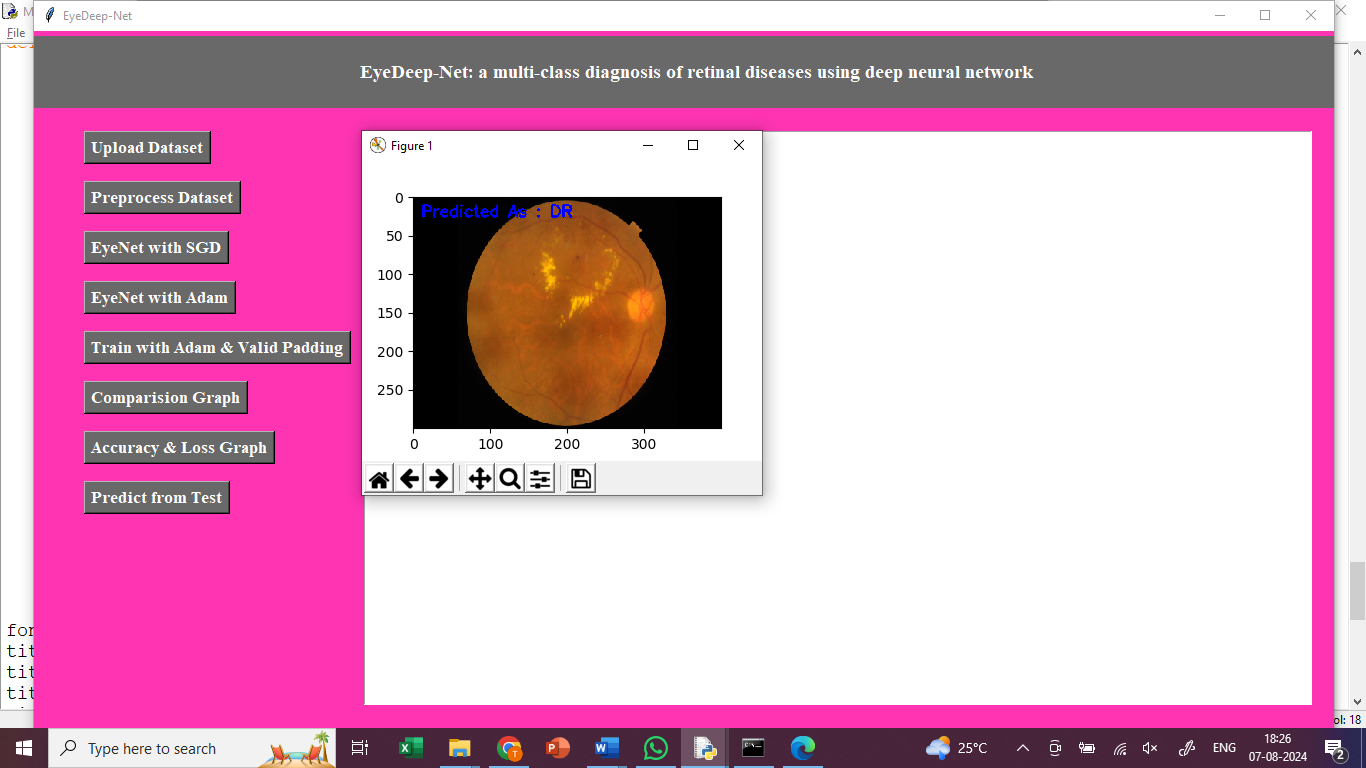
In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithm extension model got high accuracy



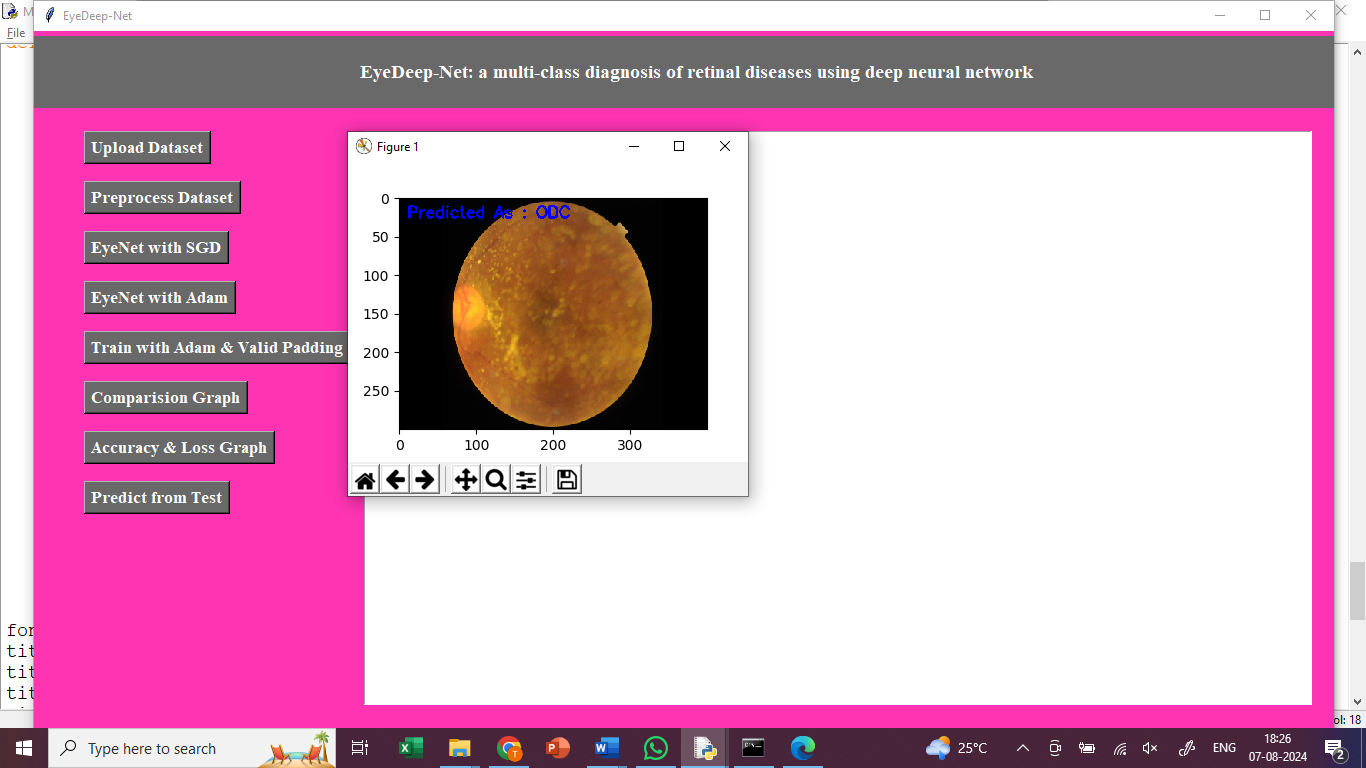
In above screen displaying training accuracy graph of propose EYENET with Adam (blue line) and SGD (green line) and extension model (black line). In above graph x-axis represents training Epoch and y-axis represents accuracy and in above graph with each increasing epoch accuracy got increase for all 3 algorithms



In above screen, click on predict from test data button, upload test image to predict



In above screen calling predict function with test image path and in blue colour text we can see predicted output as retinal disease



In above screen can see predicted output from other test images

**8. CONCLUSION:**

The classification of the different retinal disorders is addressed by presenting a CNN model based on deep learning. EyeNet, a dataset containing 32 various retinal diseases, is the basis for the model’s implementation. The proposed model is trained on different epochs to test the model’s accuracy. Initially, the model was trained at 10 epochs and achieved 95% validation accuracy; then, at 15 epochs model again achieved 95% validation accuracy with 0.0279 validation loss which varies in both cases. The model’s total performance is much superior to that of other models considered to be state of the art. There is a possibility that the categorization of retinal diseases might benefit from the model that has been provided. Regular model updates and retraining using new data will continue to enhance its performance in the future. This will be achieved by leveraging the advancements in deep learning techniques and the increasing availability of diverse retinal disease datasets.

**9. REFERENCES:**

[1] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, ‘‘Dermatologist-level classification of skin cancer with deep neural networks,’’ Nature, vol. 542, no. 7639, pp. 115–118, Feb. 2017.

[2] K. Shankar, A. R. W. Sait, D. Gupta, S. K. Lakshmanaprabu, A. Khanna, and H. M. Pandey, ‘‘Automated detection and classification of fundus diabetic retinopathy images using synergic deep learning model,’’ Pattern Recognit. Lett., vol. 133, pp. 210–216, May 2020.

[3] R. Arunkumar and P. Karthigaikumar, ‘‘Multi-retinal disease classification by reduced deep learning features,’’ Neural Comput. Appl., vol. 28, no. 2, pp. 329–334, Feb. 2017.

[4] T. Shanthi and R. S. Sabeenian, ‘‘Modified Alexnet architecture for classification of diabetic retinopathy images,’’ Comput. Electr. Eng., vol. 76, pp. 56–64, Jun. 2019.

[5] S. Farsiu, S. J. Chiu, R. V. O’Connell, F. A. Folgar, E. Yuan, J. A. Izatt, and C. A. Toth, ‘‘Quantitative classification of eyes with and without intermediate age-related macular degeneration using optical coherence tomography,’’ Ophthalmology, vol. 121, no. 1, pp. 162–172, Jan. 2014.

[6] R. F. Mullins, S. R. Russell, D. H. Anderson, and G. S. Hageman, ‘‘Drusen associated with aging and age-related macular degeneration contain proteins common to extracellular deposits associated with atherosclerosis, elastosis, amyloidosis, and dense deposit disease,’’ FASEB J., vol. 14, no. 7, pp. 835–846, May 2000.

[7] Y. Kanagasingam, A. Bhuiyan, M. D. Abràmoff, R. T. Smith, L. Goldschmidt, and T. Y. Wong, ‘‘Progress on retinal image analysis for age related macular degeneration,’’ Prog. Retinal Eye Res., vol. 38, pp. 20–42, Jan. 2014.

[8] D. S. Kermany, ‘‘Identifying medical diagnoses and treatable diseases by image-based deep learning,’’ Cell, vol. 172, no. 5, pp. 1122–1131, Feb. 2018.

[9] M. M. M. S. Fathy and M. T. Mahmoudi, ‘‘A classified and comparative study of edge detection algorithms,’’ in Proc. Int. Conf. Inf. Technol., Coding Comput., Apr. 2002, pp. 117–120.

[10] C.-H. H. Yang, J.-H. Huang, F. Liu, F.-Y. Chiu, M. Gao, W. Lyu, M. D. I.-H. Lin, and J. Tegner, ‘‘A novel hybrid machine learning model for auto-classification of retinal diseases,’’ 2018, arXiv:1806.06423.

[11] M. B. Jabra, A. Koubaa, B. Benjdira, A. Ammar, and H. Hamam, ‘‘COVID19 diagnosis in chest X-rays using deep learning and majority voting,’’ Appl. Sci., vol. 11, no. 6, p. 2884, Mar. 2021.

[12] S. Guefrechi, M. B. Jabra, A. Ammar, A. Koubaa, and H. Hamam, ‘‘‘Deep learning based detection of COVID-19 from chest X-ray images,’’ Multimedia Tools Appl., vol. 80, no. 2021, pp. 31803–31820.

[13] W. Boulila, A. Ammar, B. Benjdira, and A. Koubaa, ‘‘Securing the classification of COVID-19 in chest X-ray images: A privacy-preserving deep learning approach,’’ in Proc. 2nd Int. Conf. Smart Syst. Emerg. Technol. (SMARTTECH), May 2022, pp. 220–225.

[14] O. Perdomo, H. Rios, F. J. Rodríguez, S. Otálora, F. Meriaudeau, H. Müller, and F. A. González, ‘‘Classification of diabetes-related retinal diseases using a deep learning approach in optical coherence tomography,’’ Comput. Methods Programs Biomed., vol. 178, pp. 181–189, Sep. 2019.

[15] G. Mahendran, M. Periyasamy, S. Murugeswari, and N. K. Devi, ‘‘Analysis on retinal diseases using machine learning algorithms,’’ Mater. Today, Proc., vol. 33, pp. 3102–3107, Jan. 2020.

[16] S. J. Kim, K. J. Cho, and S. Oh, ‘‘Development of machine learning models for diagnosis of glaucoma,’’ PLoS ONE, vol. 12, no. 5, May 2017, Art. no. e0177726.

[17] P. G. Subin and P. Muthukannan, ‘‘Optimized convolution neural network based multiple eye disease detection,’’ Comput. Biol. Med., vol. 146, Jul. 2022, Art. no. 105648.

[18] M. Subramanian, M. S. Kumar, V. E. Sathishkumar, J. Prabhu, A. Karthick, S. S. Ganesh, and M. A. Meem, ‘‘Diagnosis of retinal diseases based on Bayesian optimization deep learning network using optical coherence tomography images,’’ Comput. Intell. Neurosci., vol. 2022, pp. 1–15, Apr. 2022.

[19] R. Sarki, K. Ahmed, H. Wang, Y. Zhang, and K. Wang, ‘‘Convolutional neural network for multi-class classification of diabetic eye disease,’’ EAI Endorsed Trans. Scalable Inf. Syst., vol. 9, no. 4, p. e5, 2022.

[20] D. Marín, A. Aquino, M. E. Gegundez-Arias, and J. M. Bravo, ‘‘A new supervised method for blood vessel segmentation in retinal images by using gray-level and moment invariants-based features,’’ IEEE Trans. Med. Imag., vol. 30, no. 1, pp. 146–158, Jan. 2011.

[21] X. You, Q. Peng, Y. Yuan, Y.-M. Cheung, and J. Lei, ‘‘Segmentation of retinal blood vessels using the radial projection and semi-supervised approach,’’ Pattern Recognit., vol. 44, nos. 10–11, pp. 2314–2324, Oct. 2011.

[22] G. B. Kande, P. V. Subbaiah, and T. S. Savithri, ‘‘Unsupervised fuzzy based vessel segmentation in pathological digital fundus images,’’ J. Med. Syst., vol. 34, no. 5, pp. 849–858, Oct. 2010.

[23] M. A. Palomera-Perez, M. E. Martinez-Perez, H. Benitez-Perez, and J. L. Ortega-Arjona, ‘‘Parallel multiscale feature extraction and region growing: Application in retinal blood vessel detection,’’ IEEE Trans. Inf. Technol. Biomed., vol. 14, no. 2, pp. 500–506, Mar. 2010.

[24] P. Gowsalya and S. Vasanthi, ‘‘Segmentation and classification of features in retinal images,’’ in Proc. Int. Conf. Commun. Signal Process., Apr. 2014, pp. 1869–1873.

[25] A. Das, R. Giri, G. Chourasia, and A. A. Bala, ‘‘Classification of retinal diseases using transfer learning approach,’’ in Proc. Int. Conf. Commun. Electron. Syst. (ICCES), Jul. 2019, pp. 2080–2084