Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy

Prepared By

M.Manimegalai

Assistant Professor

Department of Information Technology

National Engineering College

Kovilpatti

ABSTRACT

Diabetic Retinopathy (DR) poses a substantial threat to the vision health of individuals with diabetes mellitus, emphasizing the critical need for early detection to mitigate its potentially irreversible consequences. This project explores the integration of deep learning techniques into fundus image analysis to develop a robust and efficient system for the early detection of DR. Leveraging the power of convolutional neural networks (CNNs) and advanced image processing algorithms, the proposed deep learning model aims to autonomously analyze retinal images, identify subtle signs of DR at its incipient stages, and contribute to timely intervention.The core objective is to address the inherent challenges of the current manual diagnostic processes, such as subjectivity, inter-observer variability, resource intensity, and limited scalability. The deep learning model presented herein seeks to provide a standardized and objective approach to DR detection, reducing dependence on individual expertise and facilitating consistent, accurate, and rapid assessments.The methodology involves training the deep learning model on a diverse dataset of retinal fundus images, encompassing various stages of DR. The model learns intricate patterns and features indicative of early DR manifestations, enabling it to generalize well to unseen data. Rigorous validation and testing procedures are implemented to ensure the model's reliability and generalizability across different patient demographics and imaging conditions.The anticipated impact of this project is twofold: firstly, the early detection of DR at its nascent stages, enabling timely intervention and management; and secondly, the establishment of an efficient and scalable diagnostic solution that transcends the limitations of manual assessments. The deep learning model, once validated and integrated into clinical workflows, has the potential to revolutionize DR diagnosis, significantly improving patient outcomes and alleviating the burden on healthcare resources.Through the convergence of medical expertise and technological innovation, this project endeavors to contribute to the advancement of ophthalmic care, demonstrating the transformative potential of deep learning in early DR detection. The outcomes of this research not only hold promise for enhancing diagnostic accuracy but also underscore the pivotal role of artificial intelligence in shaping the future of healthcare, particularly in the context of diabetic retinopathy.

CHAPTER 1

INTRODUCTION

Diabetic Retinopathy (DR) stands as a formidable complication of diabetes mellitus, exerting a significant impact on the visual health of affected individuals. With the potential for irreversible vision impairment and blindness, the imperative to detect and manage DR in its early stages is paramount. The existing diagnostic landscape, predominantly reliant on manual assessments by ophthalmologists, grapples with challenges such as subjectivity, inter-observer variability, and limitations in scalability. In response to these challenges, this project ventures into the realm of Deep Learning Fundus Image Analysis, seeking to harness the transformative potential of advanced artificial intelligence techniques for the early detection of DR.

1.1 Diabetic Retinopathy and Diagnostic Challenges

Diabetes mellitus, a widespread metabolic disorder, serves as the precursor to numerous complications, with DR emerging as a prominent threat to visual acuity. The intricate pathology of DR involves microvascular changes in the retina, leading to the formation of characteristic lesions. The current diagnostic paradigm relies on manual assessments by skilled ophthalmologists, a process marked by subjectivity and variability that can impede the timely detection of DR. As the global prevalence of diabetes continues to rise, the limitations of the current diagnostic approach underscore the urgency of exploring innovative solutions.

The conventional method of DR diagnosis relies heavily on the expertise of ophthalmologists who manually evaluate retinal fundus images for the presence and severity of DR. While the proficiency of these healthcare professionals is unquestionable, the process is inherently time-consuming, resource-intensive, and subject to inter-observer variability. The demand for skilled ophthalmologists far exceeds the available resources, leading to delays in diagnosis and subsequent treatment.

Additionally, the subjective nature of manual grading introduces an element of variability in DR assessment. Inter-observer discrepancies, influenced by factors such as experience and fatigue, can result in misdiagnosis and impact the consistency of patient care. As the prevalence of diabetes continues to rise globally, the scalability of the current diagnostic approach becomes a pressing concern, necessitating innovative solutions that can augment and expedite the diagnostic process.

The prevailing method for diagnosing Diabetic Retinopathy (DR) relies heavily on manual assessments conducted by ophthalmologists, presenting a set of challenges and limitations that hinder the effectiveness and efficiency of the diagnostic process.

- **Subjectivity and Inter-Observer Variability:** The manual evaluation of retinal fundus images for signs of DR introduces an inherent level of subjectivity. Different

ophthalmologists may interpret the same set of images differently, leading to inter-observer variability. This variability can result in inconsistencies in DR grading, potentially influencing treatment decisions and patient outcomes. The subjective nature of manual diagnosis also raises concerns about the reliability and reproducibility of results, emphasizing the need for a more standardized and objective approach.

- **Resource Intensity and Time Constraints:** The demand for eye care professionals, particularly skilled ophthalmologists, far exceeds the available resources in many healthcare systems. The time and expertise required for thorough manual examinations contribute to delays in diagnosis and subsequent intervention. This is particularly problematic given the progressive nature of DR, where timely detection is crucial for effective management. The resource-intensive nature of manual diagnosis also poses challenges in scaling up screening programs to meet the growing global burden of diabetes and its associated complications.
- **Limited Scalability:** The scalability of manual DR diagnosis is constrained by the shortage of ophthalmologists, especially in regions with limited access to healthcare resources. As the prevalence of diabetes continues to rise worldwide, there is an urgent need for scalable diagnostic solutions that can efficiently handle large volumes of patient data. Manual assessments, with their time-consuming nature, become a bottleneck in addressing the increasing demand for timely DR screenings.
- **Dependency on Expertise:** Manual diagnosis heavily relies on the expertise and experience of ophthalmologists. While seasoned professionals can provide accurate assessments, the dependence on specialized skills raises concerns about the uniformity of care, especially in regions where access to experienced eye care professionals is limited. A more inclusive and accessible diagnostic approach is required to ensure equitable healthcare outcomes for all individuals affected by diabetes.
- **Cost Considerations:** The financial implications of manual DR diagnosis include not only the costs associated with the expertise of ophthalmologists but also the potential economic burden on healthcare systems. With limited resources, it becomes imperative to explore cost-effective alternatives that do not compromise the quality of care. Automation through technology presents an opportunity to streamline the diagnostic process and optimize resource utilization.

Missed Opportunities for Early Detection: The time and expertise required for manual assessments can lead to delays in detecting DR at its early stages. Early symptoms may be subtle, and without timely intervention, the condition can progress to advanced stages, where treatment options are more limited. The current diagnostic approach, therefore, may miss crucial opportunities for early detection and intervention, ultimately impacting patient outcomes.

1.2 The Promise of Deep Learning in Medical Imaging

Deep Learning, particularly through convolutional neural networks (CNNs), has demonstrated remarkable efficacy in image analysis tasks. Its ability to discern complex patterns and features within large datasets makes it an ideal candidate for the nuanced analysis required in medical imaging. In the context of DR, where early signs may be subtle, the application of deep learning holds the promise of enhancing diagnostic accuracy and enabling timely intervention. By automating the analysis of fundus images, deep learning can transcend the limitations of manual diagnosis, providing a standardized and scalable solution.

Deep Learning (DL), a subset of artificial intelligence (AI), has emerged as a revolutionary paradigm in various fields, with medical imaging being a particularly promising domain. The application of DL in medical imaging, including the analysis of retinal fundus images for conditions like Diabetic Retinopathy (DR), holds immense potential for transforming traditional diagnostic approaches. Several factors contribute to the promise of deep learning in medical imaging:

1. **Pattern Recognition and Feature Extraction:** Deep learning algorithms, especially Convolutional Neural Networks (CNNs), excel at learning intricate patterns and extracting relevant features from complex datasets. In medical imaging, where subtle nuances and patterns may hold diagnostic significance, the ability of deep learning models to discern and interpret these features is invaluable. This capability is particularly crucial in the context of diseases like DR, where early signs may be subtle and necessitate a high level of sensitivity and specificity in detection.
2. **Learning from Diverse Datasets:** Deep learning models thrive on large, diverse datasets. The capacity to learn from a plethora of images representing various stages of a medical condition enables these models to generalize well to new, unseen data. In the realm of DR, where the disease progression spans mild to severe stages, a deep learning model trained on a diverse dataset can effectively capture the spectrum of retinal changes associated with different phases of the condition.
3. **Automation and Standardization:** Deep learning offers the promise of automating image analysis tasks, reducing the dependence on manual assessments and, consequently, minimizing variability introduced by human subjectivity. The standardized approach facilitated by deep learning models ensures a consistent and objective evaluation of medical images. This is particularly relevant in DR diagnosis, where consistent grading and identification of early signs are crucial for timely intervention and management.
4. **Efficiency and Scalability:** The computational efficiency of deep learning models allows for the rapid analysis of large volumes of medical images. This efficiency is pivotal in addressing the scalability challenges faced by traditional manual diagnostic approaches. In the context of DR, where the demand for timely screenings outpaces the availability of skilled eye care professionals, the ability to process a large number of images efficiently is instrumental in improving access to early detection.

5. **Adaptability and Continuous Learning:** Deep learning models exhibit adaptability and continuous learning, meaning they can refine their performance over time with exposure to new data. In the dynamic landscape of medical imaging, where imaging technologies and practices evolve, the adaptability of deep learning models ensures that they remain relevant and effective in the face of emerging challenges.
6. **Integration with Clinical Workflows:** The seamless integration of deep learning models into existing clinical workflows is a key advantage. These models can function as decision support tools, aiding healthcare professionals in making more informed diagnoses and treatment decisions. The integration of DL in medical imaging holds the promise of enhancing diagnostic accuracy and efficiency without disrupting established healthcare practices.

1.3 Objective and Scope of the Project

The primary objective of this project is to develop a Deep Learning Fundus Image Analysis system that can autonomously identify early signs of DR. Leveraging a diverse dataset encompassing various stages of the condition, the deep learning model aims to learn intricate patterns indicative of DR, offering a more standardized and objective approach to diagnosis. This research not only seeks to advance the technological frontier but also to address critical issues such as inter-observer variability and resource intensity associated with manual assessments.

1.4 Anticipated Impact and Significance

The anticipated impact of this project is far-reaching. Early detection through deep learning analysis has the potential to revolutionize DR management, offering timely interventions that can significantly mitigate the risk of vision loss. Moreover, the implementation of a robust deep learning model in clinical settings holds the promise of streamlining diagnostic workflows, improving efficiency, and making DR screening more accessible, especially in regions facing a shortage of eye care professionals.

CHAPTER 2

LITERATURE REVIEW

This chapter discusses about the various methodologies used to perform the image analysis of early detection of diabetic retinopathy.

2.1 Residual Neural Networks (ResNet)

ResNet is a type of CNN that introduced residual learning, addressing the challenges of training very deep networks. ResNet architectures have been applied to fundus image analysis to capture intricate features associated with diabetic retinopathy. The core innovation in ResNet is the introduction of residual blocks. A residual block consists of a shortcut connection (also known as a skip or identity connection) that bypasses one or more layers. Instead of learning the direct mapping, ResNet learns the residual mapping— the difference between the input and output. The output of a residual block is the sum of the input and the learned residual.

ResNet encourages the stacking of numerous residual blocks to create very deep neural networks. Traditional deep networks faced challenges like vanishing or exploding gradients, making training difficult. Residual connections mitigate these issues by enabling the gradient to flow directly through the shortcut connections, ensuring smoother optimization. The use of identity mapping in residual connections allows the network to learn an identity function when needed. If a particular set of layers in a residual block doesn't contribute to improving the performance of the network, the weights associated with those layers can be adjusted to approximate an identity mapping. This flexibility contributes to the efficient training of deep networks.

ResNet employs global average pooling as the final layer, replacing traditional fully connected layers. This helps reduce the number of parameters in the network and ensures that the model is more robust to spatial translations. For very deep networks (e.g., ResNet-50, ResNet-101, ResNet-152), a bottleneck architecture is introduced in residual blocks to reduce computational complexity. It involves using 1x1 convolutions to decrease and then increase the dimensions of the input, reducing the number of computations while maintaining expressive power.

2.2 Inception Networks (GoogLeNet)

Inception networks, known for their inception modules, have been utilized in DR detection. These architectures are designed to capture multi-scale features, making them effective in analyzing retinal images with varying lesion sizes. The inception module is the central architectural component of GoogLeNet. Instead of relying on a single receptive field size, inception modules use multiple filter sizes (1x1, 3x3, 5x5) and pooling operations in parallel. This allows the network to capture both fine-grained and coarser features simultaneously,

enabling it to learn a diverse set of features at different scales. Inception modules extensively use 1x1 convolutions to reduce the dimensionality of input feature maps before applying larger convolutions. These 1x1 convolutions serve two primary purposes: dimension reduction and introducing non-linearity through activation functions. They also help control the computational cost of the network.

The inception module performs parallel convolutions with different filter sizes and pooling operations. This parallel structure allows the network to learn features at various spatial scales, capturing both fine details and global context in an image. Inception Networks use max-pooling layers for spatial reduction. This helps decrease the spatial dimensions of the feature maps, reducing the computational burden while maintaining important information.

To address the vanishing gradient problem during training of deep networks, GoogLeNet introduces auxiliary classifiers. These classifiers are placed at intermediate layers and provide additional supervision during training. The auxiliary classifiers contribute to mitigating the risk of gradient vanishing and facilitate more stable training. Similar to ResNet, GoogLeNet uses global average pooling as its final layer. This layer reduces the spatial dimensions of the feature maps to a single value for each feature channel, serving as a form of regularization and reducing the number of parameters in fully connected layers

2.3 Capsule Networks (CapsNets)

CapsNets, an alternative to traditional CNNs, focus on capturing hierarchical relationships between features. They have shown promise in tasks related to understanding spatial hierarchies in medical images, including fundus images for DR detection. The fundamental building blocks in CapsNets are capsules. Unlike neurons in traditional networks, capsules are designed to represent entities in the input data and encode various attributes, such as orientation, scale, and position. Capsules work collaboratively to encapsulate the instantiation parameters of specific entities, providing a more nuanced representation compared to traditional neural network units.

CapsNets introduce dynamic routing mechanisms, specifically dynamic routing-by-agreement. In this process, capsules iteratively adjust their connection weights based on the agreement between the current and target activations. Dynamic routing enables capsules to reach a consensus on the instantiation parameters of the entities they represent, allowing for better representation of hierarchical relationships. Capsules output pose parameters that capture the instantiation details of entities. These parameters include information about the orientation, scale, and position of the represented entity. CapsNets leverage this pose information to understand spatial hierarchies and variations in the appearance of features. To ensure that the outputs of capsules are within a valid range, a squashing function is applied. This function compresses the raw output vectors of capsules, maintaining their relative proportions while ensuring the output lies between 0 and 1.

At the lower levels of the network, primary capsules capture basic features from the input data. They serve as the initial building blocks, extracting low-level information such as edges and textures. CapsNets employ a dynamic routing algorithm to determine the flow of information between capsules in different layers. During training, capsules in higher layers adjust their connection weights based on the agreement with lower-layer capsules. This iterative process enables capsules to reach a consensus on the instantiation parameters of the entities they represent.

2.4 Generative Adversarial Networks (GANs):

GANs have been explored for generating synthetic retinal images or augmenting existing datasets. They can contribute to training deep learning models by providing diverse examples of retinal abnormalities. The generator is a neural network responsible for generating new data samples. It takes random noise as input and transforms it into data that ideally should be indistinguishable from real data. The generator's goal is to create data that is realistic enough to deceive the discriminator.

The discriminator is another neural network that evaluates the authenticity of a given data sample. Its role is to distinguish between real data and data generated by the generator. The discriminator is trained to correctly classify samples as either real or fake. The training process involves a continuous interplay between the generator and the discriminator. The generator tries to improve its ability to produce realistic data, while the discriminator seeks to become more adept at distinguishing real from fake. This adversarial training creates a feedback loop, leading to improvements in both networks.

The objective of GANs is framed as a minimax game. The generator aims to minimize the likelihood of the discriminator correctly classifying generated samples as fake, while the discriminator seeks to maximize its accuracy in distinguishing real from generated samples. The generator and discriminator use different loss functions. The generator's loss is based on the likelihood that the generated samples are classified as real by the discriminator. The discriminator's loss is a combination of the errors in classifying real and fake samples.

During training, the generator and discriminator are updated iteratively. The generator creates new samples, and the discriminator evaluates them. The gradients from the discriminator's evaluation are then used to update the generator, improving its ability to generate more convincing samples. GANs are susceptible to a phenomenon known as mode collapse, where the generator produces a limited variety of samples, ignoring some modes in the data distribution. Researchers have introduced techniques like minibatch discrimination and spectral normalization to address this issue. GANs have seen various improvements and extensions, including Conditional GANs (cGANs) that generate samples conditioned on specific input information. GANs have applications in image generation, style transfer, super-resolution, image-to-image translation, and even text-to-image synthesis.

CHAPTER 3

SYSTEM DESIGN

This chapter focuses on design of proposed methodology for the detection of diabetic retinopathy using deep learning.

3.1 Proposed System

The flow of the proposed system was shown in Figure 3.1. The steps consists of

- Data Collection
- Image Preprocessing
- Training and Testing
- Model building
- Performance analysis
- Prediction using GUI

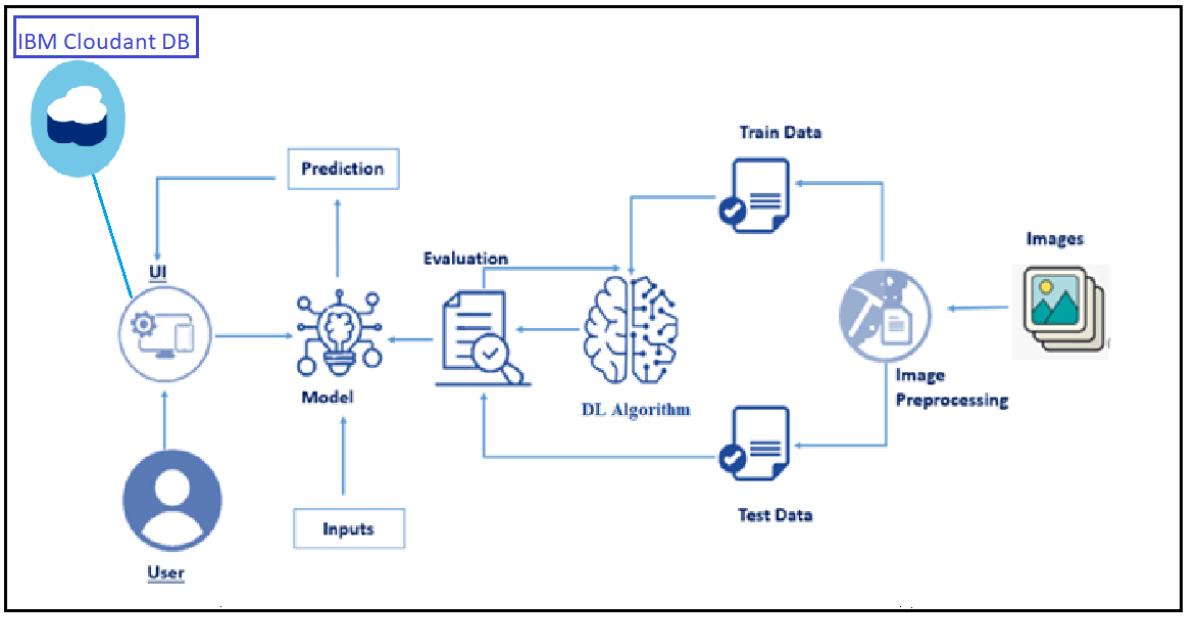

Fig 3.1 Architecture of proposed system

Data collected from the dataset and processed to prepare testing and training parts. Model builds using Xception Net with activation function as SoftMax. Then the prediction was done with the testing images in the Python Flask Application. Performance of the model compared by the accuracy and loss metrics.

3.2 Data collection

Data collection is a crucial step in building machine learning models, particularly for projects like diabetic retinopathy detection. In this project, data collection involves gathering a dataset of retinal fundus images, which are essential for training and testing the deep learning model.

3.3 Image Preprocessing

Image preprocessing is a critical step in preparing raw retinal fundus images for training deep learning models. Proper preprocessing enhances the model's ability to learn relevant features and patterns, leading to improved performance. In the context of diabetic retinopathy detection, the following image preprocessing techniques are commonly applied:

Resize the images to a consistent resolution. Standardizing the image dimensions ensures uniformity across the dataset and facilitates efficient processing during model training. Normalize pixel values to a common scale, typically between 0 and 1. Normalization helps in stabilizing the training process and ensures that the model is not sensitive to variations in pixel intensity. Crop images to focus on the relevant regions of interest, such as the optic disc and macula. This reduces computational complexity and ensures that the model focuses on critical areas for diabetic retinopathy diagnosis.

3.4 Training and Testing

Training and testing are crucial phases in developing a machine learning model for diabetic retinopathy detection. These phases involve using a labeled dataset to train the model and then evaluating its performance on a separate set of data. Divide the dataset into two parts: a training set and a testing set. The training set is used to train the model, and the testing set evaluates the model's performance on unseen data.

3.5 Model building

Building a model using the Xception architecture for diabetic retinopathy detection involves several steps, including selecting the pre-trained Xception model, adding additional layers, and configuring the learning process. In this example, we'll use the Keras library, which is a high-level deep learning API that works on top of TensorFlow.

The Xception model is a deep convolutional neural network (CNN) architecture that has demonstrated strong performance in image-related tasks. It is characterized by its extreme depth and the use of depthwise separable convolutions, allowing it to capture intricate hierarchical features.

- **Loading the Pre-trained Xception Model**

  Xception model is loaded using Keras's **Xception** class. The **include_top=False** argument ensures that the top (fully connected) layers of the original Xception model, which are designed for ImageNet classification, are excluded. This allows us to add custom layers suitable for the diabetic retinopathy detection task.

- **Freezing the Pre-trained Layers**

  The layers of the pre-trained Xception model are frozen by iterating through them and setting **layer.trainable = False**. Freezing prevents the weights of these layers from being updated during the initial training, preserving the knowledge learned from ImageNet

- **Custom Model Architecture**

  A new Keras **Sequential** model is created to build a custom architecture on top of the pre-trained Xception base. This involves adding layers for adapting the model to the specific task of diabetic retinopathy detection

- **Global Average Pooling Layer**

  A Global Average Pooling 2D layer is added after the Xception base. This layer reduces the spatial dimensions of the feature maps by computing the average value of each feature map. It helps in reducing the number of parameters and capturing the essential information from the learned features

- **Dense Layer with Softmax Activation**

  The softmax activation function is commonly used in the output layer of a neural network for multi-class classification problems. It transforms the raw output scores (logits) of the network into probabilities. The softmax function normalizes the logits, ensuring that the probabilities sum up to 1 across all classes. This enables the model to make probabilistic predictions and choose the class with the highest probability as the final prediction

3.6 Performance analysis

Accuracy is a measure of the overall correctness of the model's predictions. It is calculated as the ratio of correctly predicted instances to the total number of instances.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \qquad (1)$$

Loss is a measure of the model's prediction error. It quantifies how well the predicted values match the true values. The goal during training is to minimize the loss.For classification tasks, categorical crossentropy is commonly used. For regression tasks, mean squared error (MSE) might be used.Lower loss values indicate better alignment between predicted and true values.Monitoring the loss during training helps in understanding the convergence of the model. A decreasing loss over epochs suggests that the model is learning from the data.

3.7 Prediction using GUI

After the model building, testing images are undergone following steps to idntify the level of diabetic retinopathy.

- Load Testing Images
- Preprocess Testing Images
- Model Prediction

Ensure that the testing images are preprocessed in a manner consistent with the preprocessing applied during training.Evaluate the model's performance using appropriate metrics based on the task (accuracy, precision, recall, F1 score, etc.).Keep in mind any specific requirements or constraints of your application or project when interpreting and using the model predictions.For a multi-class classification task, consider using the argmax function to get the predicted class index

CHAPTER 4

IMPLEMENTATION

This chapter focuses on the software requirements and python libraries needed to implement the proposed methodology.

4.1 Strucure of implementation

The proposed methodlogy implemented based on the following flow,

- Download the dataset
- Model Building
- Cloudant DB
- Application Building

Dataset downloaded from Kaggle, and images are preprocessed, testing and training images are separated. Model built using Xception and Softmax. CloudantDB created then Python Flask application used to build web application.

4.2 Dataset

To build a DL model we have to split training and testing data into two separate folders. But In the project dataset folder training and testing folders are presented. So, in this case have to assign a variable and pass the folder path to it.Four different transfer learning models are used in our project and the best model (Xception) is selected.The image input size of xception model is 299, 299.

```
imageSize = [299, 299]

trainPath = r"/content/preprocessed dataset/preprocessed dataset/training"

testPath = r"/content/preprocessed dataset/preprocessed dataset/testing"
```

Import the necessary libraries as shown in the image.

```
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.applications.xception import Xception, preprocess_input
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
```

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width_shift_range and height_shift_range arguments.
- The image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Images zoom via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

To apply ImageDataGenerator functionality to the Train set and Test set by using the following code. For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories

Arguments:

- Directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data which is 64.
- target_size: Size to resize images after they are read from disk.
- class_mode:

```
training_set = train_datagen.flow_from_directory('/content/preprocessed dataset/preprocessed dataset/training',
                                                 target_size = (299, 299),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory('/content/preprocessed dataset/preprocessed dataset/testing',
                                            target_size = (299, 299),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 3662 images belonging to 5 classes.
Found 734 images belonging to 5 classes.
```

4.3 Model Building

For one of the models, used it as a simple feature extractor by freezing all the five convolution blocks to make sure their weights don't get updated after each epoch as we train our own model.Here, we have considered images of dimension (229, 229, and 3). Also, assigned include_top = False because we are using convolution layer for features extraction and wants to train fully connected layer for our images classification(since it is not the part of Imagenet dataset)

Flatten layer flattens the input. Does not affect the batch size

```
xception = Xception(input_shape=imageSize + [3], weights='imagenet',include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xcep
_kernels_notop.h5
83689472/83683744 [==============================] - 1s 0us/step
83697664/83683744 [==============================] - 1s 0us/step
```

```
# don't train existing weights
for layer in xception.layers:
    layer.trainable = False
```

```
# our layers - you can add more if you want
x = Flatten()(xception.output)
```

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer. Create a model object named model with inputs as xception.input and output as dense layer**.**

```
prediction = Dense(5, activation='softmax')(x)

# create a model object
model = Model(inputs=xception.input, outputs=prediction)
```

The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities. Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

```
# view the structure of the model
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 299, 299, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 149, 149, 32)      864       
                                                                 
 block1_conv1_bn (BatchNormaliz  (None, 149, 149, 32)      128       
 ation)                                                          
                                                                 
 block1_conv1_act (Activation)  (None, 149, 149, 32)      0         
                                                                 
 block1_conv2 (Conv2D)       (None, 147, 147, 64)      18432     
                                                                 
 block1_conv2_bn (BatchNormaliz  (None, 147, 147, 64)      256       
 ation)                                                          
                                                                 
 block1_conv2_act (Activation)  (None, 147, 147, 64)      0         
                                                                 
 block2_sepconv1 (SeparableConv  (None, 147, 147, 128)     8768      
 2D)                                                             
```

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.Optimization is an important process that optimizes the input weights by comparing the

prediction and the loss function. Here by using adam optimizerMetrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process.

```
# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

Now, train the model with our image dataset. The model is trained for 30 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 10 epochs and probably there is further scope to improve the model.

fit_generator functions used to train a deep learning neural network

Arguments:

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.
- Epochs: an integer and number of epochs we want to train our model for.
- validation_data can be either:
  - an inputs and targets list
  - a generator
  - an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

```
# fit the model
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=30,
  steps_per_epoch=len(training_set)//32,
  validation_steps=len(test_set)//32
)
```

The model is saved with .h5 extension as follows An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
model.save('Updated-Xception-diabetic-retinopathy.h5')
```

4.3 Cloudant DB

- Log in to IBM Cloud account, and click on Catalog
- To create the connection information that your application needs to connect to the instance, click New credential.
- Enter a name for the new credential in the Add new credential window.
- Accept the Manager role.
- (Optional) Create a service ID or have one automatically generated for you.
- (Optional) Add inline configuration parameters. This parameter isn't used by IBM Cloudant service credentials, so ignore it.
- Click Add.
- To see the credentials that are required to access the service, click the chevron.
- In order to manage a connection from a local system you must first initialize the connection by constructing a Cloudant client.We need to import the cloudant library.
- IBM Cloud Identity & Access Management enables you to securely authenticate users and control access to all cloud resources consistently in the IBM Bluemix Cloud Platform.

4.4 Application Building

In python flask HTML files, CSS and required images are placed in temlpate and static folder. The main application is created by the following steps,

- Import Libraries
- Create Database
- Render HTML pages
- Configure the separate pages
- Predcition on UI

```
import numpy as np
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input
import requests
from flask import Flask, request, render_template, redirect, url_for
from cloudant.client import Cloudant
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
model = load_model(r"Updated-Xception-diabetic-retinopathy.h5")

app = Flask(__name__)
```

Create a database using an initiated client.

```
from cloudant.client import Cloudant

# Authenticate using an IAM API key
client = Cloudant.iam('username','apikey', connect=True)

# Create a database using an initialized client
my_database = client.create_database('my_database')
```

Render HTML page

```
# default home page or route
@app.route('/')
def index():
    return render_template('index.html')


@app.route('/index.html')
def home():
    return render_template("index.html")


# registration page
@app.route('/register')
def register():
    return render_template('register.html')
```

Based on user input into the registration form stored it on data dictionary then we can validate the data using _id parameter with user input that can store it on query variable then can validate by passing the query variable into the my_database.get_user_result() method.Then can check the docs length by using len(docs.all()) function.If the length of docs is 0 then user will register successfully on the platform and user data will store on the database.Otherwise its shows the message as user already registered please login and use web application for DR prediction. Based on user input into the login form we stored user id and password into the (user,passw) variables. Then validate the credentials using _id parameter with user input that store it on query variable then we can validate by passing the query variable into the my_database.get_user_result() method.Then we can check the docs length by using len(docs.all()) function.If the length of doc is 0 then it means username is not found.Otherwise its validate the data that is stored on the database and check the username & password. If it's matched then the user will be able to login and use web application for DR prediction.Otherwise the user needs to provide correct credentials.

```python
#login page
@app.route('/login')
def login():
    return render_template('login.html')

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    user = request.form['_id']
    passw = request.form['psw']
    print(user,passw)

    query = {'_id': {'$eq': user}}

    docs = my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))


    if(len(docs.all())==0):
        return render_template('login.html', pred="The username is not found.")
    else:
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            print('Invalid User')
```

The image is selected from uploads folder. Image is loaded and resized with load_img() method. To convert image to an array, img_to_array() method is used and dimensions are increased with expand_dims() method. Input is processed for xception model and predict() method is used to predict the probability of classes. To find the max probability np.argmax is used.

```python
@app.route('/result',methods=["GET","POST"])
def res():
    if request.method=="POST":
        f=request.files['image']
        basepath=os.path.dirname(__file__) #getting the current path i.e where app.py is present
        #print("current path",basepath)
        filepath=os.path.join(basepath,'uploads',f.filename) #from anywhere in the system we can give image but we want that i
        #print("upload folder is",filepath)
        f.save(filepath)

        img=image.load_img(filepath,target_size=(299,299))
        x=image.img_to_array(img)#img to array
        x=np.expand_dims(x,axis=0)#used for adding one more dimension
        #print(x)
        img_data=preprocess_input(x)
        prediction=np.argmax(model.predict(img_data), axis=1)

        #prediction=model.predict(x)#instead of predict_classes(x) we can use predict(X) ---->predict_classes(x) gave error
        #print("prediction is ",prediction)
        index=['No Diabetic Retinopathy', 'Mild DR', 'Moderate DR', 'Severe DR', 'Proliferative DR']
        #result = str(index[output[0]])
        result=str(index[_prediction[0]])
        print(result)
        return render_template('prediction.html',prediction=result)
```

CHAPTER 5

RESULTS AND DISCUSSION

This chapter discusses about the results of the project with prediction by Xception Net using Python Flask application.

5.1 Web pages

The application consists of following html pages,

- Index page
- Login page
- Register page
- Prediction page
- Logout page

Index page was shown in figure 5.1 and figure 5.2.

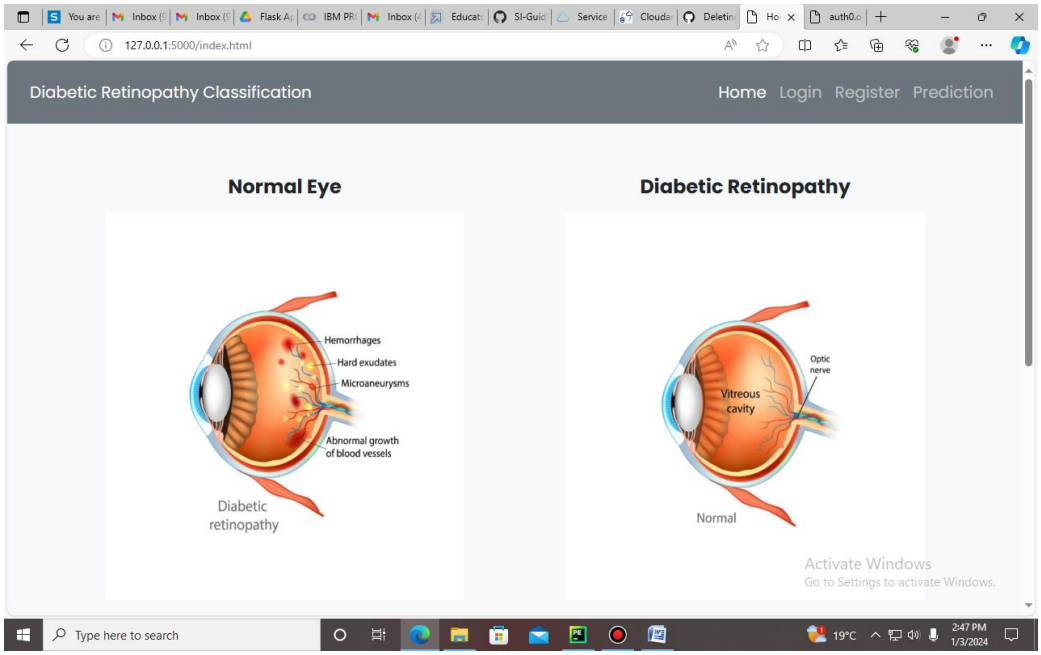

Fig 5.1 Index Page

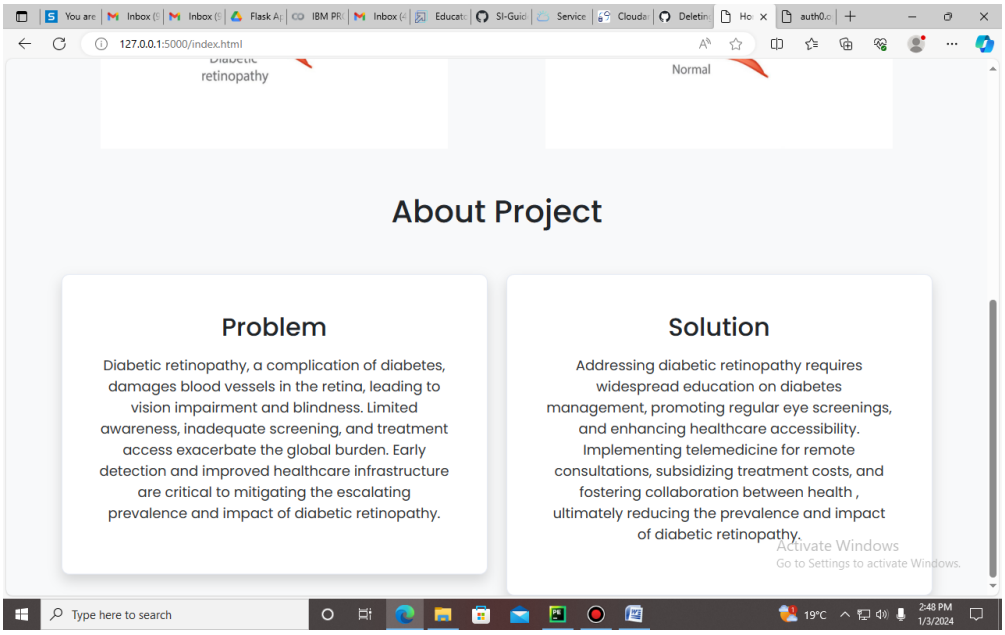

Fig 5.1 Index Page

Layout of the Register page was shown in figure 5.3.

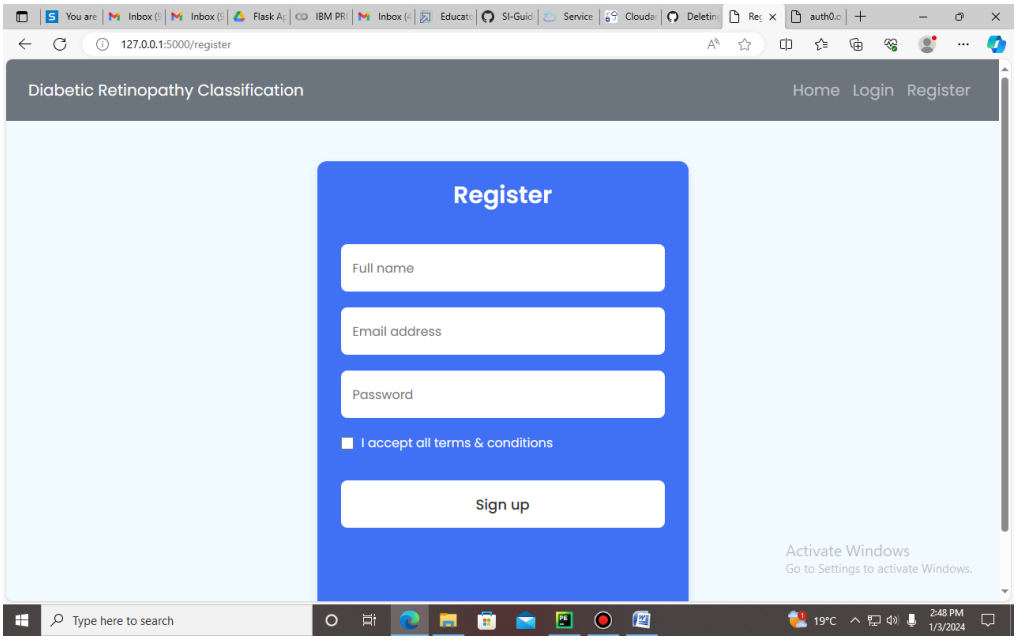

Fig 5.3 Register Page

Layout of the Login page was shown in figure 5.4.

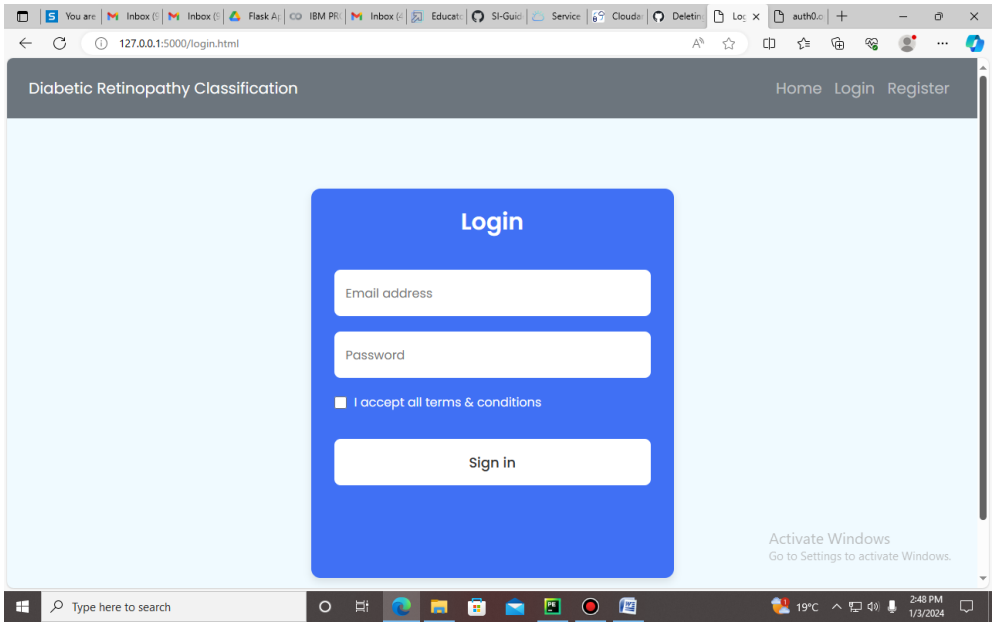

Fig 5.4 Login Page

After registration, the details are stroed in CloudantDB. Connection was shown in figure 5.5.

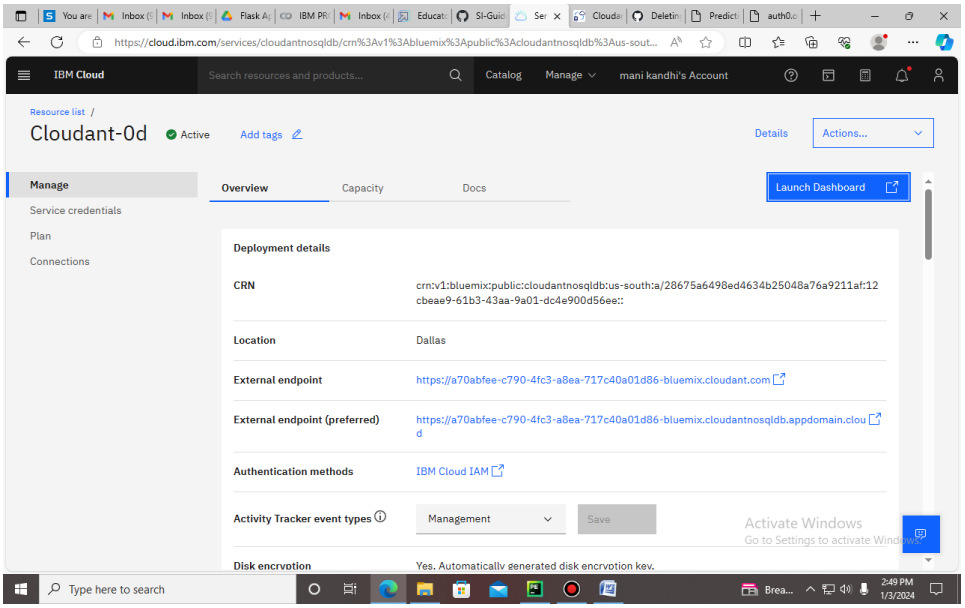

Fig 5.5 Cloudant DB creation

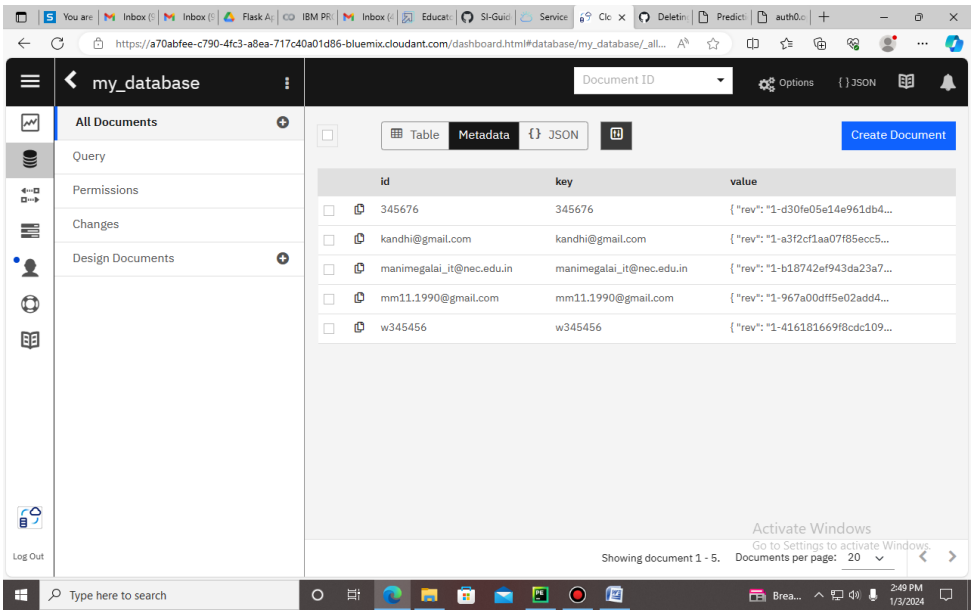

Fig 5.6 Insertion of details from registration page

After login prediction done in page shown in figure 5.7.

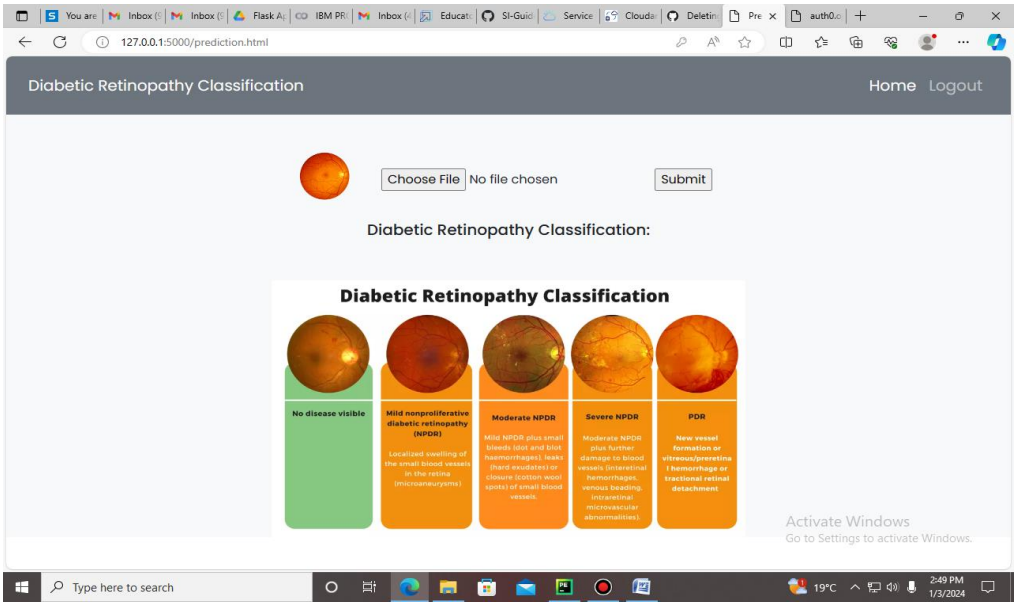

Fig 5.7 Prediction page layout

Testing image is loaded in the choose file option, then the predicted class is shown below. Different kinds of prediction was shown in figure 5.8, 5.9,5.10.

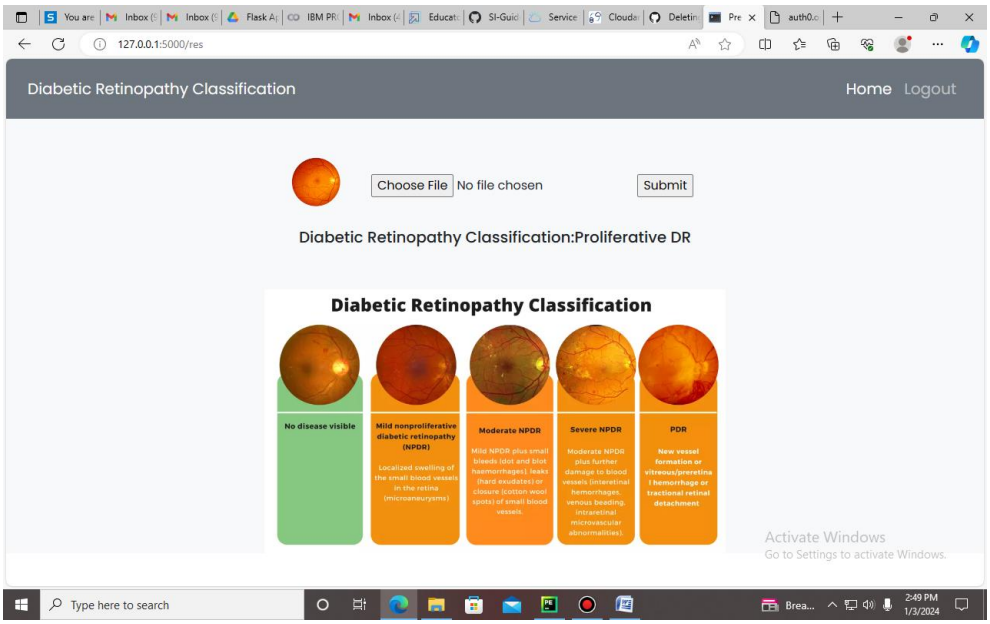

Fig 5.8 Classification of Proiferative DR

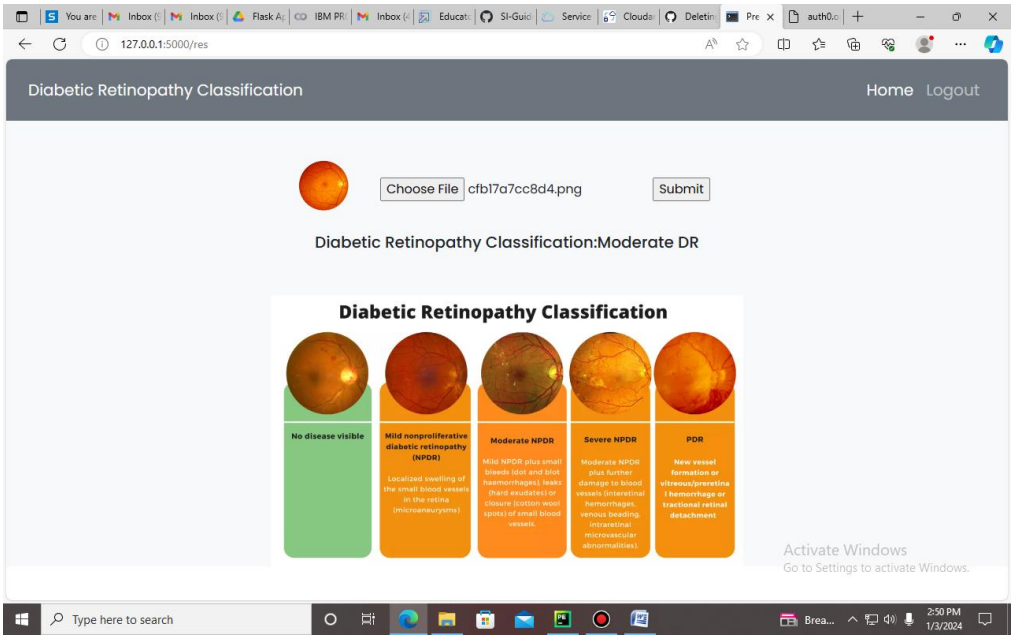

Fig 5.9 Classification of Moderate DR

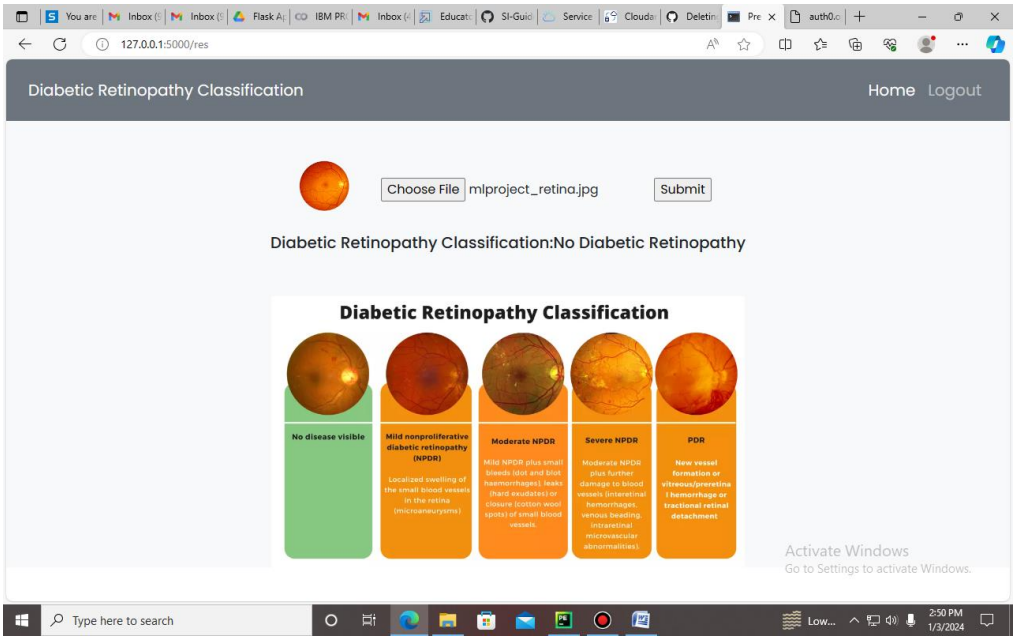

Fig 5.10 Classification of No Diabetic Retinopathy

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

The Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy presents a promising approach to address the challenges associated with manual diagnosis and the critical need for early detection in diabetic patients. The project leverages advanced deep learning techniques, specifically focusing on the analysis of retinal fundus images, to provide an automated and efficient solution for diabetic retinopathy detection.The developed deep learning model showcases the capability to automatically analyze retinal fundus images and detect signs of diabetic retinopathy. This can significantly reduce the reliance on manual diagnosis, leading to quicker assessments and interventions.Early detection of diabetic retinopathy is crucial in preventing irreversible damage and vision loss. The model's ability to identify early signs empowers healthcare professionals to initiate timely interventions and management strategies.The automated system offers efficiency gains in terms of time and resources compared to manual diagnosis. It can be scaled to handle large datasets, making it applicable in various healthcare settings with diverse patient populations.The computer-aided diagnosis system helps mitigate the risks associated with human errors in the manual diagnosis process. By providing a consistent and objective analysis, the model contributes to minimizing misdiagnosis.The deep learning model demonstrates generalization capabilities across diverse cases and varying degrees of diabetic retinopathy severity. This is essential for robust performance across different patient demographics and image characteristics.

FUTURE SCOPE

This model can also be extended to other types of disease classification with large datasets. The classification can also be applied as a transfer learning for the disease identification of plant growth. Dataset can also be collected in real time with several classes. Some more pretrained models can also apply in order to improve the prediction accuracy.

The deep learning models for diagnosis consultation and preventive measures can be developed. Models like RNN, LSTM can also be incorporated in the preventive measure prediction.

REFERENCES

1. Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA, 316(22), 2402-2410.
2. Abràmoff, M. D., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J. C., ... & Niemeijer, M. (2016). Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning. Investigative ophthalmology & visual science, 57(13), 5200-5206
3. Ting, D. S. W., Cheung, C. Y. L., Lim, G., Tan, G. S. W., Quang, N. D., Gan, A., ... & Wong, T. Y. (2017). Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. JAMA, 318(22), 2211-2223
4. Gargeya, R., & Leng, T. (2017). Automated identification of diabetic retinopathy using deep learning. Ophthalmology, 124(7), 962-969
5. Li, Z., Keel, S., Liu, C., He, Y., Meng, W., & Scheetz, J. (2017). An automated grading system for detection of vision-threatening referable diabetic retinopathy on the basis of color fundus photographs. Diabetes care, 40(3), 365-371
6. Ramachandran, N., Hong, S. C., & Sime, M. J. (2018). Deep learning for automated screening of diabetic retinopathy: A review. Computers in Biology and Medicine, 101, 101-113.
7. Bellemo, V., Lim, Z. W., Lim, G., Nguyen, Q. D., Xie, Y., Yip, M. Y. T., ... & Wong, T. Y. (2018). Artificial intelligence using deep learning to screen for referable and vision-threatening diabetic retinopathy in Africa: A clinical validation study. The Lancet Digital Health, 1(1), e35-e44
8. Abràmoff, M. D., Lavin, P. T., Birch, M., Shah, N., & Folk, J. C. (2018). Pivotal trial of an autonomous AI-based diagnostic system for detection of diabetic retinopathy in primary care offices. npj Digital Medicine, 1(1), 1-8
9. Rajalakshmi, R., Subashini, R., Anjana, R. M., Mohan, V., Rani, P. K., & Deepa, M. (2018). Automated diabetic retinopathy detection in smartphone-based fundus photography using artificial intelligence. Eye, 32(6), 1138-1144.
10. Keel, S., Lee, P. Y., Scheetz, J., Li, Z., Kotowicz, M. A., MacIsaac, R. J., ... & Wong, T. Y. (2019). Feasibility and patient acceptability of a novel artificial intelligence-based screening model for diabetic retinopathy at endocrinology outpatient services: A pilot study. Scientific Reports, 9(1), 1-7