**Deep Learning Fundus Image Analysis For Early Detection Of Diabetic Retinopathy**

**Introduction:**

Diabetic Retinopathy (DR) is a common complication of diabetes mellitus, which causes lesions on the retina that affect vision. If it is not detected early, it can lead to blindness. Unfortunately, DR is not a reversible process, and treatment only sustains vision. DR early detection and treatment can significantly reduce the risk of vision loss. The manual diagnosis process of DR retina fundus images by ophthalmologists is time, effort and cost-consuming and prone to misdiagnosis unlike computer-aided diagnosis systems. Transfer learning has become one of the most common techniques that has achieved better performance in many areas, especially in medical image analysis and classification. We used Transfer Learning techniques like Resnet50,Xception V3 that are more widely used as a transfer learning method in medical image analysis and they are highly effective.

**Technical Architecture:**

A diagram of a medical procedure

Description automatically generated

**Prerequisites:**

# To complete this project, you must require the following software’s, concepts, and packages

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using

Anaconda watch the video

Link: [Click here to](https://www.youtube.com/watch?v=5mDYijMfSzs&feature=emb_logo) watch the video

# To build Machine learning models you must require the following packages

* **Numpy**:
  + It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations
* **Scikit-learn:**
  + It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy
* **Flask:**

Web framework used for building Web applications

* **Python packages:**
  + - open anaconda prompt as administrator
    - Type “pip install numpy” and click enter. o Type “pip install pandas” and click enter. o Type “pip install scikit-learn” and click enter.
    - Type “pip install tensorflow==2.3.2” and click enter. o Type “pip install keras==2.3.1” and click enter.
    - Type “pip install Flask” and click enter.

# ● Deep Learning Concepts

* **Transfer Learning:** A technique where a pre-trained neural network model is used as a starting point for a new, related task. Instead of training a model from scratch, which can be computationally expensive and time-consuming, transfer learning leverages the knowledge gained from solving one task to improve performance on a different but related task.
* **Flask:** Flask is a popular Python web framework, meaning it is a third-party

Python library used for developing web applications.

[**Flask Basics**](https://www.youtube.com/watch?v=lj4I_CvBnt0)

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

**Project Objectives:**

By the end of this project you will:

* Know fundamental concepts and techniques of Convolutional Neural Network.
* Gain a broad understanding of image data.
* Know how to pre-process/clean the data using different data preprocessing techniques.
* know how to build a web application using the Flask framework.

**Project Flow:**

* The user interacts with the UI (User Interface) to choose the image.
* The chosen image analyzed by the model which is integrated with flask application.
* TL Models analyze the image, then prediction is showcased on the Flask UI.

To accomplish this, we have to complete all the activities and tasks listed below

* Data Collection. o Create Train and Test Folders.
* Data Preprocessing.
* Import the ImageDataGenerator library o Configure ImageDataGenerator class o ApplyImageDataGenerator functionality to Trainset and Testset
* Model Building o Import the model building Libraries o Initializing the model o Adding Input Layer o Adding Hidden Layer
* Adding Output Layer o Configure the Learning Process o Training and testing the model o Save the Model
* Application Building o Create an HTML file o Build Python Code

**Project Structure:**

Create a Project folder which contains files as shown below

A screenshot of a computer program

Description automatically generated

* The Dataset folder contains the training and testing images for training our model.
* We are building a Flask Application that needs HTML pages stored in the **templates** folder and a python script **app.py** for server side scripting
* we need the model which is saved and the saved model in this content is **inception-diabetic.h5.**
* ● templates folder contains base.html,index.html pages.

# Milestone 1: Data Collection

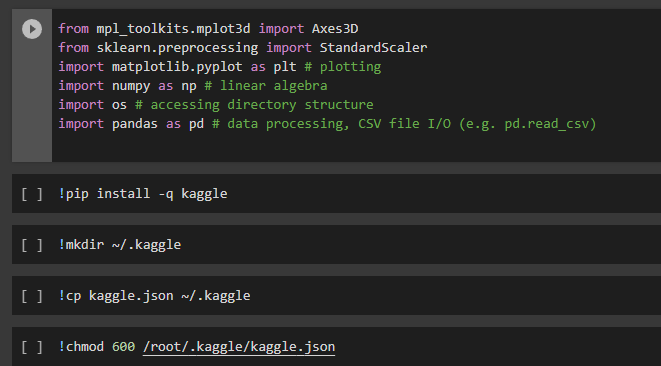
For our data collection we are using the dataset in the Kaggle.

**Download the Dataset-** <https://www.kaggle.com/datasets/arbethi/diabetic-retinopathy-level-detection?select=preprocessed+dataset>

# Activity 1:Loading the dataset into the Google colab:

We are going to clone the Kaggle dataset on colab. Kindly refer to this - <https://www.analyticsvidhya.com/blog/2021/06/how-to-load-kaggle-datasets-directly-into-google-colab/>

* Clone kaggle in google colab



* Download the dataset

A screenshot of a computer

Description automatically generated

* Unzip the dataset

A screenshot of a computer

Description automatically generated

# Activity 2: Create Training And Testing Path:

To build a TL 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, we just 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 (Resnet50) is selected.

The image input size of xception model is 224, 224.

A screen shot of a computer

Description automatically generated

# Milestone 2: Data Pre-Processing:

# In this milestone we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

# Activity 1: Importing The Libraries

# import the necessary libraries as shown in the image.

# A screen shot of a computer Description automatically generated

# Activity 2: Apply keras preprocessing functionality To Train Set And Test Set

Let us apply keras preprocessing functionality to the Train set and Test set by using the following code. For Training set using image\_dataset\_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. I have stored the directory in the variable ‘trainPath’ for training\_data and ‘testPath’ for validation\_data respectively.
* batch\_size: Size of the batches of data which is  16.
* image\_size: Size to resize images after they are read from disk.
* shuffle: To shuffle the data and set as True.
* Seed: used to set the random seed for shuffling the data during the creation of the image datasets
* Subset: used to specify whether the dataset being created is a subset of the entire dataset.
* Validation\_split: used to specify the fraction of the dataset that should be reserved for validation. Set it 0.01 for our case.

A computer screen shot of a program

Description automatically generated

# Milestone 3: Model Building:

Now it's time to build our Transfer Learning. We are using Resnet50 model to our project.

# Activity 1: Pre-Trained TL Model As A Feature Extractor:

For one of the models, we will use 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 image\_shape=(224,224,3).

 Also, we have assigned include\_top = False because we are using the convolution layer for features extraction and want to train a fully connected layer for our images classification(since it is not the part of Imagenet dataset)

pooling = ‘avg’

classes=5

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

A computer screen shot of a program code

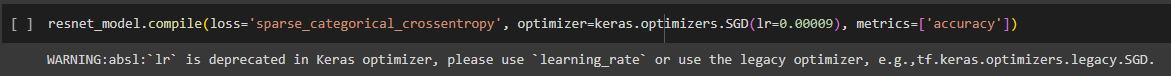
Description automatically generated

# Activity 2: Configure The Learning Process:

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 we are using adam optimizer

Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process



# Activity 3: Train The Model:

Now, let us train our 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

                      - 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.

# A screen shot of a computer program Description automatically generated

# A screenshot of a computer Description automatically generated

A screenshot of a computer program

Description automatically generated

# Activity 4: Saving the model:

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.

A screenshot of a computer program

Description automatically generated

# Activity 5: Testing the model:

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.

Load the saved model using load\_model

Taking an image as input and checking the results

By using the model we are predicting the output for the given input image

The predicted class index name will be printed here.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated A screenshot of a computer screen

Description automatically generated A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

# Milestone 4: Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Garbage and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the “Image” button, the next page is opened where the user chooses the image and predicts the output.

**Activity 1 : Create HTML Pages o** We use HTML to create the front end part of the web page.

* Here, we have created 3 HTML pages- home.html, intro.html, and upload.html **o** home.html displays the home page.
* Intro.html displays an introduction about the project
* upload.html gives the emergency alert

For more information regarding HTML <https://www.w3schools.com/html/>

* We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

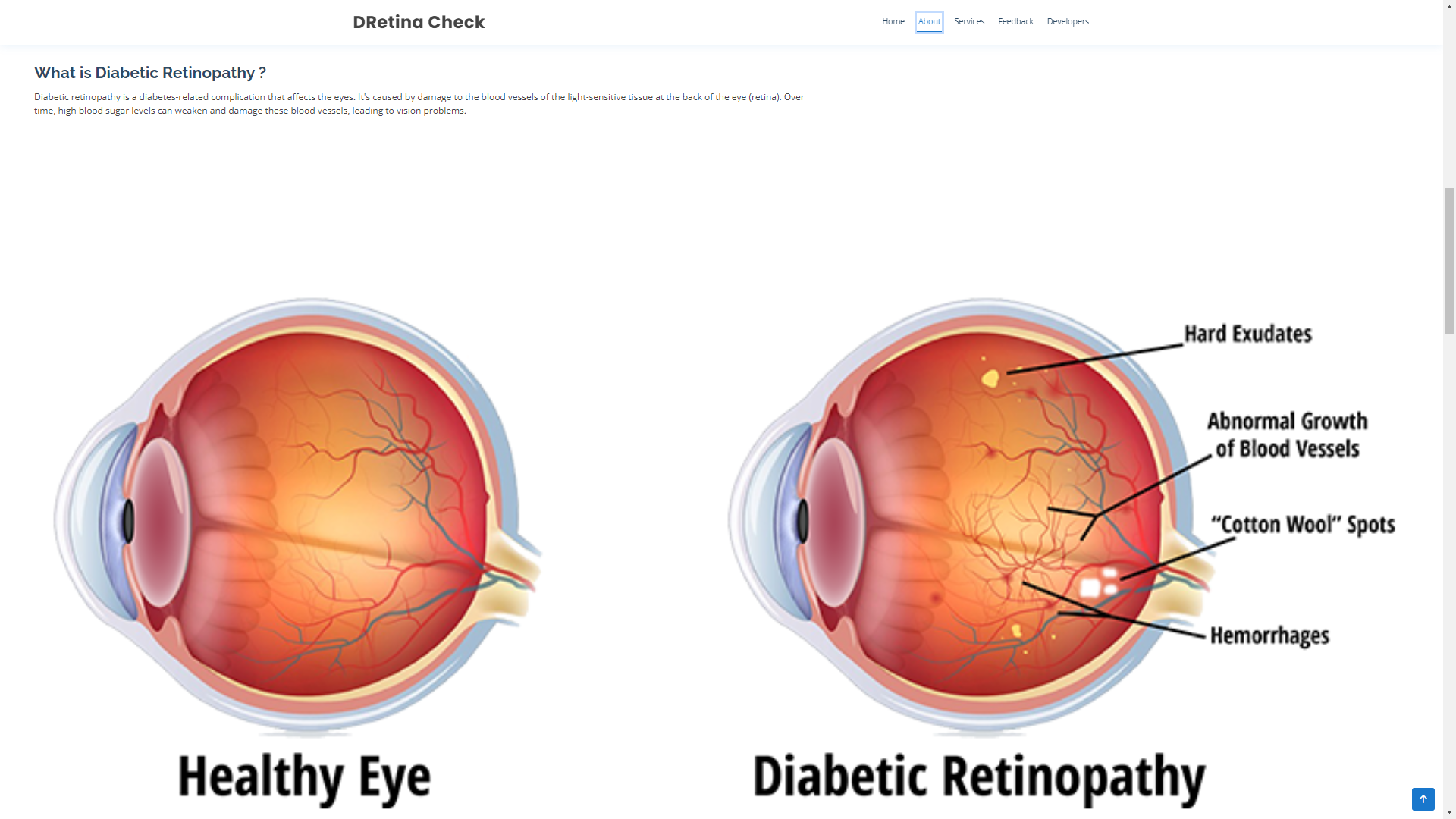
**○ Link :**[CSS](https://www.w3schools.com/css/) , [JS](https://www.w3schools.com/js/DEFAULT.asp)

# index.html looks like this

A screenshot of a computer

Description automatically generated

About us:



Services:

A screenshot of a web page

Description automatically generated

FAQ:

A screenshot of a computer

Description automatically generated

Feedback section:

A screenshot of a feedback form

Description automatically generated

Developers and footer section:

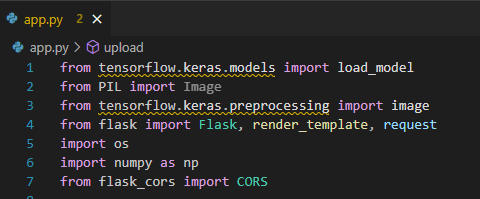
A screenshot of a computer

Description automatically generated

**Activity 2: Build python code**

# Task 1: Importing Libraries

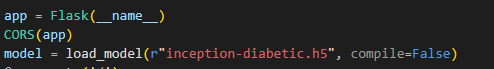
The first step is usually importing the libraries that will be needed in the program.



Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument

Pickle library to load the model file.

# Task 2: Creating our flask application and loading our model by using load\_model method



# Task 3: Routing to the html Page

Here, the declared constructor is used to route to the HTML page created earlier.

In the below example, ‘/’ URL is bound with index.html function. Hence, when the home page of a web server is opened in the browser, the html page will be rendered. Whenever you browse an image from the html page this photo can be accessed through POST or GET Method.

A screen shot of a computer

Description automatically generated

**Showcasing prediction on UI:**

A screenshot of a computer

Description automatically generated

Here we are defining a function which requests the browsed file from the html page using the post method. The requested picture file is then saved to the uploads folder in this same directory using OS library. Using the load image class from Keras library we are retrieving the saved picture from the path declared. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class.

# Finally, Run the application

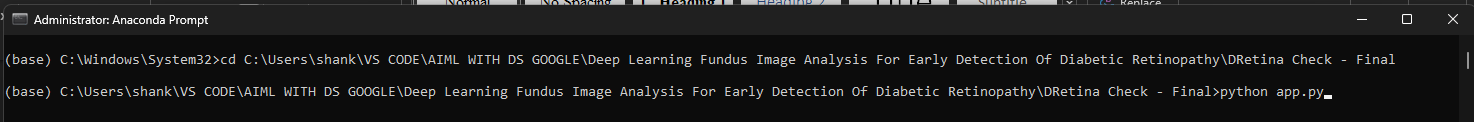
This is used to run the application in a local host.

A black background with text and numbers

Description automatically generated

# Activity 3:Run the application

* Open the anaconda prompt from the start menu.
* Navigate to the folder where your app.py resides.
* Now type “python app.py” command.
* It will show the local host where your app is running on **http://127.0.0.1.5000/**
* Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
* Enter the values, click on the predict button and see the result/prediction on the web page.



Then it will run on localhost:5000

A screen shot of a computer

Description automatically generated

Navigate to the localhost ([http://127.0.0.1:5000/](http://127.0.0.1:8000/))where you can view your web page.

**FINAL OUTPUT:**

**Output 1:**

**1:**

A screenshot of a computer

Description automatically generated

**Output 2:**

A screenshot of a computer

Description automatically generated