**DOCUMENTATION**

**DIABETIC RETINOPATHY SCREENING**

**1. Introduction**

Diabetic retinopathy is a medical condition in which damage occurs to the retina due to diabetes mellitus. It is a leading cause of blindness. Detecting diabetic retinopathy in its early stages is crucial for preventing vision loss. This project aims to develop a machine learning model to detect diabetic retinopathy using image processing and classification techniques.

**2. Authors**

* Arshad G
* Rohan S
* Siddharth Kumar Iyer
* Prakkash Manohar

**3. Environment Setup**

To run this project, you need to install the following libraries:

python

Copy code

from scipy import misc

from PIL import Image

from skimage import exposure

from sklearn import svm

import scipy

from math import sqrt, pi

from numpy import exp

from matplotlib import pyplot as plt

import numpy as np

import glob

import matplotlib.pyplot as pltss

import cv2

from matplotlib import cm

import pandas as pd

import pywt

import tensorflow as tf

import tensorflow\_hub as hub

If you are using Google Colab, you may need to mount your Google Drive to access datasets:

python

Copy code

from google.colab import drive

drive.mount('/content/drive/')

**4. Data Pre-processing**

The pre-processing steps involve loading images, converting them to grey-scale, and performing adaptive histogram equalization.

python

Copy code

import cv2

import matplotlib.pyplot as plt

import os

# Define the path to the dataset and image

data\_dir = './diabetic-retinopathy-detection'

image\_name = 'sample\_image.jpeg'

img\_pt = os.path.join(data\_dir, 'train', image\_name)

# Read the image using OpenCV

img = cv2.imread(img\_pt)

if img is None:

print(f"Error: Unable to load image from '{img\_pt}'")

else:

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

plt.imshow(img)

plt.title('Sample Image')

plt.axis('off')

plt.show()

**5. Feature Extraction**

Feature extraction is performed using wavelet transformation. Here is an example of how to extract features:

python

Copy code

import pywt

# Perform wavelet transformation

coeffs2 = pywt.dwt2(img\_gray, 'bior1.3')

LL, (LH, HL, HH) = coeffs2

**6. Model Training**

The K-Nearest Neighbors (KNN) algorithm is used to train the model. Here is an example:

python

Copy code

from sklearn.neighbors import KNeighborsClassifier

imm\_train = np.array([[1, 2], [2, 3], [3, 4], [4, 5], [5, 6]])

y\_train = np.array([0, 0, 1, 1, 0])

neigh = KNeighborsClassifier(n\_neighbors=3)

neigh.fit(imm\_train, y\_train)

**7. Evaluation**

The model's accuracy is evaluated using the test dataset.

python

Copy code

imm\_kmean = np.array([[1, 2], [2, 3], [3, 4]])

Y = np.array([0, 1, 1])

accuracy = neigh.score(imm\_kmean, Y)

print(f"Accuracy: {accuracy \* 100:.2f}%")

**8. Visualization**

Visualization of results using heatmaps and saliency masks is an essential part of the analysis.

python

Copy code

def ShowImage(im, title, ax=None):

if ax is None:

P.figure()

P.axis('off')

P.imshow(im, cmap=P.cm.gray, vmin=0, vmax=1)

P.title(title)

def ShowHeatMap(im, title, ax=None):

if ax is None:

P.figure()

P.axis('off')

P.imshow(im, cmap='viridis')

P.title(title)

**9. Conclusion**

The model achieved a final accuracy of 94.38% using the KNN algorithm. Future work may include exploring different algorithms and enhancing the feature extraction process to improve accuracy further.