DRUG CONSUMPTION DETECTION BY EYE USING DEEP LEARNING

A PROJECT REPORT

Submitted by

K. ARCHANA
(REG. NO: 21MCR008)

K.P. ARUN HIRUTHIK
(REG. NO: 21MCR009)

P. DINESH KUMAR
(REG. NO: 21MCR022)

in partial fulfillment of the requirements
for the award of the degree
of

MASTER OF COMPUTER APPLICATIONS
DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

JANUARY 2023

DEPARTMENT OF COMPUTER APPLICATIONS
KONGU ENGINEERING COLLEGE

(Autonomous)
PERUNDURAI, ERODE – 638 060

JANUARY 2023

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"DRUG CONSUMPTION DETECTION BY EYE USING DEEP LEARNING"** is the bonafide record of project work done by **K.ARCHANA (21MCR008), K.P.ARUN HIRUTHIK (21MCR009)** and **P.DINESHKUMAR (21MCR022)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications of Anna University Chennai during the year 2022 - 2023.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

Submitted for the end semester viva- voce examination held on___________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We affirm that the project report entitled **”DRUG CONSUMPTION DETECTION BY EYE USING DEEP LEARNING”** being submitted in partialfulfillment of the requirements for the award of Master of Computer Applications is the originalwork carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

K. ARCHANA (21MCR008)

K.P. ARUNHIRUTHIK (21MCR049)

Date: **P. DINESH KUMAR (21MCR022)**

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: Name and Signature of the Supervisor with seal

[Dr.M.PYINGKODI]

ABSTRACT

Python has been used to create the project’s front end. Drug usage is a significant global economic issue that results in countless losses. This study suggests a convolutional neural network-based image processing method for identifying drugged eyes. The front-end packages already in use for this project collect information from eye images using the CNN algorithm. However, it can take some time. As a result, the suggested approach may be used to swiftly and automatically identify drugged eyes.

The acquisition of the input image, picture pre-processing, and image processing are the main processes in the suggested technique. locating the reddish regions, emphasising the afflicted regions, confirming the training set, and presenting the findings. Some eyes may go unnoticed, such those that have been socially poisoned. This technique was tried on drugged eyes at different phases, both with and without drug consumption. The technique was used to find a white region of the eye in a given input picture. Images of both drugged and unaffected eyes were presented for training purposes.

In order to choose the optimum color model for this strategy, photos were first transformed to color models. The Support Erosion technique and the Local Binary Pattern were both used in the model construction for feature extraction. This technique can detect drugged eyes with an average accuracy of 95%.

Keywords – CNN algorithm, Deep Learning, Image Segmentation, Image Pre-processing.

ACKNOWLEDGEMENT

We respect and thank our correspondent **Thiru.A.K.ILANGO BCom., MBA., LLB** and our Principal **Dr.V.BALUSAMY BE(Hons)., MTech., PhD.,** Kongu Engineering College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Professor and Head of the Department **Dr.R.THAMILSELVAN ME., Ph.D.,** Department of Computer Applications, Kongu Engineering College, for his perfect guidance and support that made this work to be completed successfully.

We would like to express our gratitude and sincere thanks to our project coordinators **Dr.L.RAHUNATHAN MCA., Ph.D.,** Associate Professor and **Mrs.S.HEMALATHA MCA.,** Assistant professor (Sr.G) and Department of Computer Applications, Kongu Engineering College, who has motivated us in all aspects for completing the project in the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Dr.M.PYINGKODI, MCA., M.Phil., Ph.D,** Assistant Professor (Sr.G), Department of Computer Applications, Kongu Engineering College for giving his valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping overwhelm in all proceedings. We bow our heart and head with heartfelt thanks to all those who thought us their warm services to succeed and achieve our work.

TABLE OF CONTENTS

CHAPTER NO.	**TITLE**	**PAGE**
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	**INTRODUCTION**	1
	1.1 IMAGE PREPROCESSING	1
	1.2 DEEP LEARNING MODEL	2
	1.3 OBJECTIVES	3
2	**LITERATURE SURVEY**	4
3	**METHODOLOGY**	10
	3.1 Proposed Work	10
	3.1.1 Image Acquisition	11
	3.1.2 Image Preprocessing	11
	3.1.3 Image Segmentation	11
	3.1.4 Applying Training Set	12
	3.1.5 Experimental Results	12
	3.2 WORKFLOW DESCRIPTION	15
	3.3 INPUT DESIGN	17
	3.4 OUTPUT DESIGN	18

4	**PROJECT IMPLEMENTATION**	19
	4.1 GRAY SCALING	20
	4.2 HISTOGRAM	21
	4.3 IMAGE PROCESSING	22
	4.4 CONVOLUTION NERUAL NETWORK	23
	4.5 IMPLEMENTATION	23
5	**CONCLUSION AND FUTURE ENHANCEMENTS**	27
	5.1 CONCLUSION	27
	5.2 SCOPE FOR FUTURE DEVELOPMENT	27
	APPENDIX	29
	REFERENCES	39

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	Block Diagram for Proposed Work	10
3.8	Detection of Drugged Eye	16
4.1	Gray-Scaling	20
4.2	Conversion of Gray Scale to Histogram	21
4.2	Pictorial Representation of Histogram	22

LIST OF ABBREVIATIONS

ABBREVATIONS	EXPLANATION
CNN	Convolutional Neural Network
GUI	Graphical User Interface
HSV	Hue, Saturation & Value
ERP	Enterprise resource planning

CHAPTER 1

INTRODUCTION

The classical approach for detection and identification of drugged eyes is based on the naked eye observation by the experts. In some developing countries, consulting experts are expensive and time consuming due to the distant locations of their availability. Automatic detection of drugged eye is essential to automatically detect the symptoms of drug consumers. Drugs can cause major losses in many industrial fields. To know what control factors to take next year to avoid losses, it is crucial to recognize what is being observed.

However, detection of defects is still problematic due to natural variability of white area of eye in different types of eye, high variance of defect types, and presence of red area. The studies of eye can be determined by apparent patterns of specific types and it is critical to monitor reddish area within a eye.

1.1 IMAGE PREPROCESSING

Image preprocessing is the process of preparing an image for further analysis or processing. In the context of detecting and identifying drugged eyes, image preprocessing may involve a variety of steps to enhance the quality of the input images and make them more suitable for subsequent processing steps. Some possible image preprocessing steps that could be applied to the images of drugged eyes include

Images may be contaminated with noise due to various sources, such as sensor noise, transmission errors, or environmental interference. Noise reduction techniques can be used to smooth out the noise and improve the overall quality of the image Variations in lighting conditions can affect the appearance of the eyes in the images. Illumination correction techniques can be used to normalize the images and remove any unwanted shading or reflections. Different cameras and lighting conditions can cause variation in the colors of the images. Color correction techniques can be used to standardize the colors across different images and remove any color casts or biases.

Depending on the requirements of the subsequent processing steps, it may be necessary to crop or resize the images to a consistent size and aspect ratio. The next step in the process may involve extracting relevant features from the images for further analysis or classification. These features could include the shape, size, and color of the eyes, as well as any reddish areas or other abnormalities.

1.2 DEEP LEARNING MODEL

Deep learning, also called neural networks, is a subset of machine learning that uses a model of computing that's very much inspired by the structure of the eye. Deep learning is already working in Google search and in image search; it allows you to image-search a term like 'hug.' It's used to getting you Smart Replies to your Gmail. It's in speech and vision. It will soon be used in machine translation, I believe." said Geoffrey Hinton, considered the Godfather of neural networks.

Deep Learning models, with their multi-level structures, as shown above, are very helpful in extracting complicated information from input images. Convolutional neural networks are also able to drastically reduce computation time by taking advantage of GPU for computation which many networks fail to utilize.

Image classification using CNN is most effective. First and foremost, we need a set of images. In this case, we take images of eyes, as our initial training data set. The most common image data input parameters are the number of images, image dimensions, number of channels, and number of levels per pixel.

1.3 OBJECTIVES

The main objectives of the research are to

- To give eye image input as well as with drug consuming can be given for finding the name of disease.
- To initiate the given input image for image processing.
- To convert the RGB image into binary format to make sure it is drugged.
- To highlight the reddish area of eye.
- To apply the training image set to find the drug consumed eye.
- To provide accurate result about the given input image.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

Over the past two decades, biometric recognition has exploded into a plethora of different applications around the globe. This proliferation can be attributed to the high levels of authentication accuracy and user convenience that biometric recognition systems afford end-users. However, in-spite of the success of biometric recognition systems, there are a number of outstanding problems and concerns pertaining to the various sub-modules of biometric recognition systems that create an element of mistrust in their use - both by the scientific community and also the public at large. Some of these problems include: i) questions related to system recognition performance, ii) security (spoof attacks, adversarial attacks, template reconstruction attacks and demographic information leakage), iii) uncertainty over the bias and fairness of the systems to all users, iv) explainability of the seemingly black-box decisions made by most recognition systems, and v) concerns over data centralization and user privacy. In this paper, we provide an overview of each of the aforementioned open-ended challenges. We survey work that has been conducted to address each of these concerns and highlight the issues requiring further attention. Finally, we provide insights into how the biometric community can address core biometric recognition systems design issues to better instill trust, fairness, and security for all **[1].**

The personal identification approaches using iris images are receiving increasing attention in the biometrics literature. Several methods have been presented in the literature and those based on the phase encoding of texture information are suggested to be the most promising. However, there has not been any attempt to combine these approaches to achieve further improvement in the performance. This paper presents a comparative study of the performance from the iris authentication using Log-Gabor, Haar wavelet, DCT and FFT based features. Our experimental results suggest that the performance from the Hear wavelet and Log-Gabor filter based phase encoding is the most promising among all the four approaches considered in this work. Therefore, the combination of these two matchers is most promising, both in terms of performance and the computational complexity. Our experimental results from the all 411 users (CASIA v3) and 224 users (IITD v1) database illustrate significant improvement in the performance which is not possible with either of these approaches individually **[2].**

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork **[3].**

Algorithms developed by the author for recognizing persons by their iris patterns have now been tested in many field and laboratory trials, producing no false matches in several million comparison tests. The recognition principle is the failure of a test of statistical independence on iris phase structure encoded by multi-scale quadrature wavelets. The combinatorial complexity of this phase information across different persons spans about 249 degrees of freedom and generates a discrimination entropy of about 3.2 b/mm/sup 2/ over the iris, enabling real-time decisions about personal identity with extremely high confidence. The high confidence levels are important because they allow very large databases to be searched exhaustively (one-to-many "identification mode") without making false matches, despite so many chances. Biometrics that lack this property can only survive one-to-one ("verification") or few comparisons. The paper explains the iris recognition algorithms and presents results of 9.1 million comparisons among eye images from trials in Britain, the USA, Japan, and Korea **[4].**

This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. A third criterion is then added to ensure that the detector has only one response to a single edge. We use the criteria in numerical optimization to derive detectors for several common image features, including step edges. On specializing the analysis to step edges, we find that there is a natural uncertainty principle between detection and localization performance, which are the two main goals. With this principle we derive a single operator shape which is optimal at any scale. The optimal detector has a simple approximate implementation in which edges are marked at maxima in gradient magnitude of a Gaussian-smoothed image. We extend this simple detector using operators of several widths to cope with different signal-to-noise ratios in the image. We present a general method, called feature synthesis, for the fine-to-coarse integration of information from operators at different scales. Finally we show that step edge detector performance improves considerably as the operator point spread function is extended along the edge **[5].**

In this paper, we present the evolution of the open source iris recognition system OSIRIS through its more relevant versions: OSIRISV2, OSIRISV4, and OSIRISV4.1. We developed OSIRIS in the framework of BioSecure Association as an open source software aiming at providing a reference for the scientific community. The software is mainly composed of four key modules, namely segmentation, normalization, feature extraction and template matching, which are described in detail for each version. A novel approach for iris normalization, based on a non geometric parameterization of contours is proposed in the latest version: OSIRISV4.1 and is detailed in particular here. Improvements in performance through the different versions of OSIRIS are reported on two public databases commonly used, ICE2005 and CASIA-IrisV4-Thousand. We note the high verification rates obtained by the last version. For this reason, OSIRISV4.1 can be proposed as a baseline system for comparison to other algorithms, this way supplying a helpful research tool for the iris recognition community **[6].**

In this paper we proposed a novel multimodal biometric approach using iris and periocular biometrics to improve the performance of iris recognition in case of non-ideal iris images. Though iris recognition has the highest accuracy among all the available biometrics, still the noises at the image acquisition stage degrade the recognition accuracy. The periocular region can act as a supporting biometric, in case the iris is obstructed by several noises. The periocular region is the part of the face immediately surrounding the eye. The approach is based on fusion of features of iris and periocular region. The approach has shown significant improvement in the performance of iris recognition. The evaluation was done on a test database created from the images of UBIRIS V2 and CASIA iris interval database. We achieved identification accuracy up to 96 % on the test database **[7].**

The primary objective of this paper is to propose a complete methodology for eye liveness detection based on pupil dynamics. This method may serve as a component of presentation attack detection in iris recognition systems, making them more secure. Due to a lack of public databases that would support this paper, we have built our own iris capture device to register pupil size changes under visible light stimuli, and registered 204 observations for 26 subjects (52 different iris), each containing 750 iris images taken every 40ms. Each measurement registers the spontaneous pupil oscillations and its reaction after a sudden increase of the intensity of visible light. The Kohn and Clynes pupil dynamics model is used to describe these changes; hence we convert each observation into a feature space defined by model parameters. To answer the question whether the eye is alive (that is, if it reacts to light changes as a human eye) or the presentation is suspicious (that is, if it reacts oddly or no reaction is observed), we use linear and nonlinear support vector machines to classify natural reaction and spontaneous oscillations, simultaneously investigating the goodness of fit to reject bad modeling. Our experiments show that this approach can achieve a perfect performance for the data we have collected. All normal reactions are correctly differentiated from spontaneous oscillations. We investigated the shortest observation time required to model the pupil reaction, and found that time periods not exceeding 3 s are adequate to offer a perfect performance **[8].**

The term robot means different things to different people. Science fiction books and movies have strongly influenced what many people expect a robot to be or what it can do. Sadly the practice of robotics is far behind this popular conception. One thing is certain though – robotics will be an important technology in this century. Products such as vacuum cleaning robots are the vanguard of a wave of smart machines that will appear in our homes and workplaces **[9].**

Shiftwork can be a risk factor for a number of different somatic and psychological health conditions, especially sleep disorders. Shift workers sleep less than dayworkers, and 20–40% of them suffer from difficulties initiating and maintaining sleep, which result in reduced capacity for work and social life. A common coping strategy might be the use of alcohol, which presents a health and safety hazard as it further impairs sleep quality and exacerbates sleepiness in the workplace. This review aimed to assess the extent of such possible connections. *Methods:* We performed a systematic search of the scientific literature on shiftwork and alcohol consumption in PubMed, PsycInfo, and Cochrane Library. Only original studies comparing shiftworkers with non-shiftworkers were included. The recommendations of the *Preferred Reporting Items of Systematic Reviews and Meta-Analyses* were followed. *Results:* Fourteen articles are included in this review. Six studies report some kind of connection between shift- or nightwork and alcohol consumption, especially as a sleep aid. Conflicting or negative results are reported by 3 studies. *Discussion:* Shiftwork, especially working at night and in rotation shifts, is associated with binge drinking disorder in different professions. The reasons for pathological consumption of alcohol can be self-medication of sleep problems or coping with stress and psychosocial problems typical for shiftwork. Nurses aged over 50 years represent one important risk group. These results can be important for preventive programs against sleep disorders, including measures other than drinking alcohol as a sleep aid in the workplace of shift workers **[10].**

SUMMARY

In the above literature review, researchers have used various techniques and deep learning algorithm to detect the eye tracking. In each work different method used to extract the feature. The most commonly used techniques are CNN and Random Forest, Logistic Regression, Decision tree.

CHAPTER 3

METHODOLOGY

3.1 PROPOSED WORK

For the drugged eye detection problem, precise image segmentation is required; otherwise the features of the non-reddish region will dominate over the features of the reddish region. In this approach CNN based image processing is preferred to detect the region of red which is the infected part only. After processing the input image, features are extracted from the processed image of the eye. Finally, training and classification are performed and the exact result has been provided.

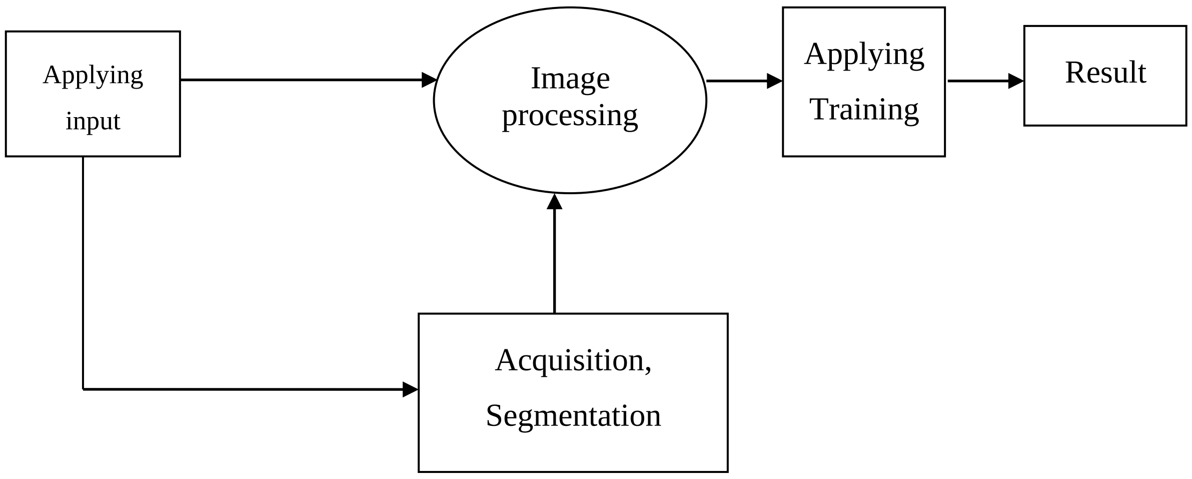


Figure 3.1 Block Diagram for Proposed Work

The above Figure 3.1 describe different methodology. Those are Image Acquisition, Image Preprocessing, Image Segmentation, Applying training dataset, Experimental results.

3.1.1 Image Acquisition

In this phase, the sample images are collected, which are required to train the classifier algorithm and build the classifier model. Reddish passion eye variety was selected to take sample images. Because the yellowish variety is widely not available to detect the drug in our country. Healthy and affected eye images were taken by using mobile phone digital camera and used for both training and testing the classifier algorithm. Images were taken in different angles, under the different environmental and lighting conditions. The standard JPG format was used to store these images. In this study, images were collected from in different regions. Eyes infected by reddish disease that had been included in collected images.

3.1.2 Image Preprocessing

After the image acquisition, image processing was done for improving the image quality. All original eye images were stored in one folder. Those images were named as we like our wish can take any value of numbers. Only horizontal images were rotated by 90 degrees and resized by 200x300 pixels. Vertical images were resized by 200x300 pixels and when the width and height of the image are same, those images were resized to 250x250 pixels. When the image size is too large, the processing task takes more time. After that, one of the noise reduction methods was used to remove the noises from images and increase the sharpness of images. Later, all preprocessed images were saved in a folder.

3.1.3 Image Segmentation

The third phase of the methodology is image segmentation. As the first step, all preprocessed images were converted into L*a*b, HSV, Grey color models and kept one in the original way (RGB). Because the identifying suitable color model for preprocessing is one of the outcomes of this research. After that, the image was converted to binary format.

This format values were clustered using the CNN algorithm. According to the algorithm used an image segmentation were done.

3.1.4 Applying Training Set

The fifth phase of the methodology is applying training set images. The segmented output were done, which were created using feature extraction. However, two image sets were created to do experiments. Preparation of those image sets is discussed here. Field expertise support was taken for the categorization of images and each image were selected from the categorized sets of an image randomly. The fifth phase of the methodology is applying training set images. The segmented output were done, which were created using feature extraction. However, two image sets were created to do experiments. Preparation of those image sets is discussed here. Field expertise support was taken for the categorization of images and each image were selected from the categorized sets of an image randomly.

3.1.5 Experimental Results

After applying the training set images, two base folders were used for identifying drugged eyes according to its accuracy. These files are called as “2 classes dataset”. Another way is counting number of reddish places to identify drug consumed eye according to its stage. This method is called as alternative method. Every training and testing time, rows of training files were shuffled randomly for increasing the accuracy of the model. Each training file was verified and tested in five times and accuracy was taken. Average of those accuracies was taken as the accuracy of each model. Using this image dataset, two types of categories were found. Such as drugged eye and not drugged eye.

3.1.6 Drugged Eye

A histogram is a graph that represents the distribution of a dataset. In the context of drugged person eye images, a histogram could be used to visualize the distribution of various features within the images, such as color intensity or shape. For example, a histogram could be used to compare the distribution of red color intensity between drugged and non-drugged eyes, or to identify any distinctive shape or texture patterns that are more common in drugged eyes. By analyzing the histograms of drugged and non-drugged eye images, it may be possible to identify features that are indicative of drug use and use them to develop a classifier for detecting drugged eyes. Ex: Figure 1,2,3.

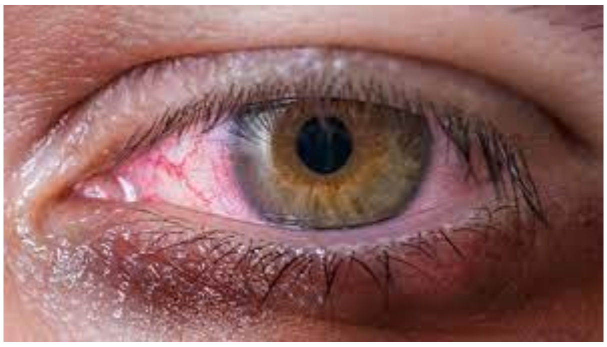


Figure 3.2 Drugged Eye Image No.1

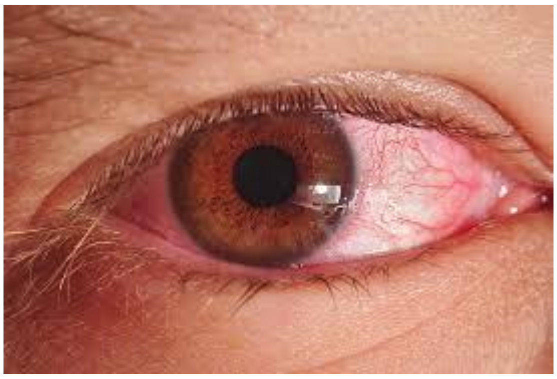


Figure 3.3 Drugged Eye Image No.2

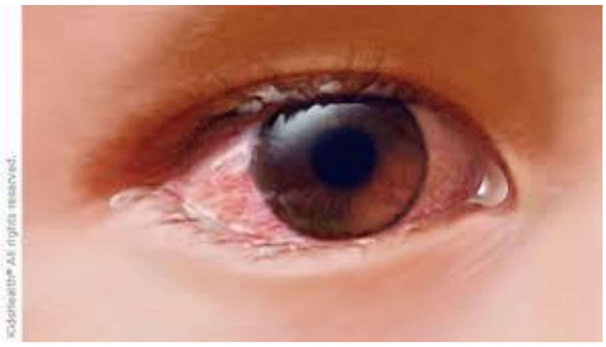


Figure 3.4 Drugged Eye Image No.3

3.1.7 Non-Drugged Eye

A histogram is a graph that represents the distribution of a dataset. In the context of non-drugged person eye images, a histogram could be used to visualize the distribution of various features within the images, such as color intensity or shape. For example, a histogram could be used to compare the distribution of red color intensity between drugged and non-drugged eyes, or to identify any distinctive shape or texture patterns that are more common in non-drugged eyes. By analyzing the histograms of drugged and non-drugged eye images, it may be possible to identify features that are indicative of drug use and use them to develop a classifier for detecting drugged eyes. Ex: Figure 4,5,6.

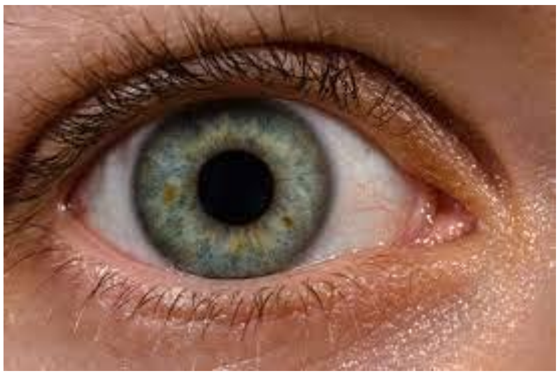


Figure 3.5 Non-Drugged Eye Image No.1

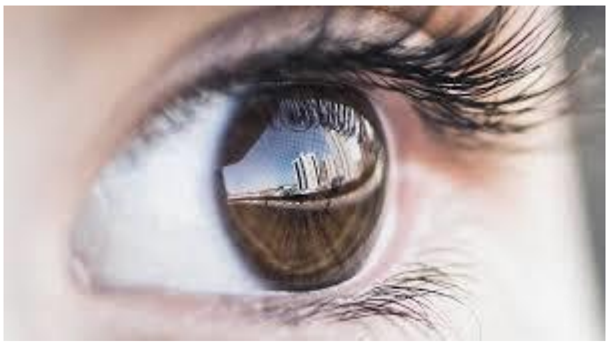


Figure 3.6 Non-Drugged Eye Image No.2

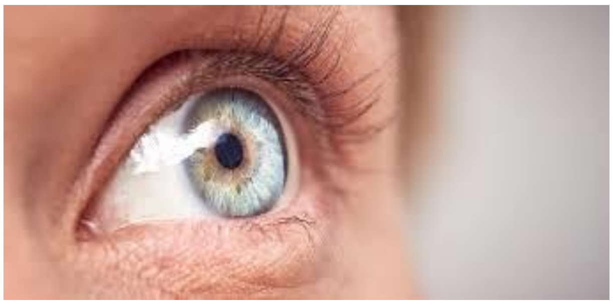


Figure 3.7 Non-Drugged Eye Image No.3

3.2 WORKFLOW DESCRIPTION

The input images are an "eye image" and a "training image" which could be used to train a model to detect specific features in the eye image. The process begins with clustering. Clustering is an unsupervised machine learning technique that groups similar data points together. In this context, it could be used to group similar pixels together in the eye image. It could be used to identify specific regions of interest in the eye image like Pupil and Iris. Edge detection/Segmentation: It's a technique used to identify boundaries of objects in an image. It can be used to identify the boundaries of the iris and pupil in the eye image. It could be used Canny Edge Detection algorithm to achieve this. Red Area Identification: It is a technique that could be used to identify areas of redness in the eye image. This could be useful for detecting drug-induced eye dilation, which can cause the iris to appear reddish. This could be used a color thresholding or color space conversion to identify red areas.

View Detected Result: This step allows to view the final output after applying all the processing steps. It could be a binary image where the detected regions are highlighted in white and the rest is black. In summary, the script is likely processing an eye image to detect regions of interest and drug-induced changes in the eye, such as dilation and redness, using a combination of clustering, edge detection, and color thresholding techniques.

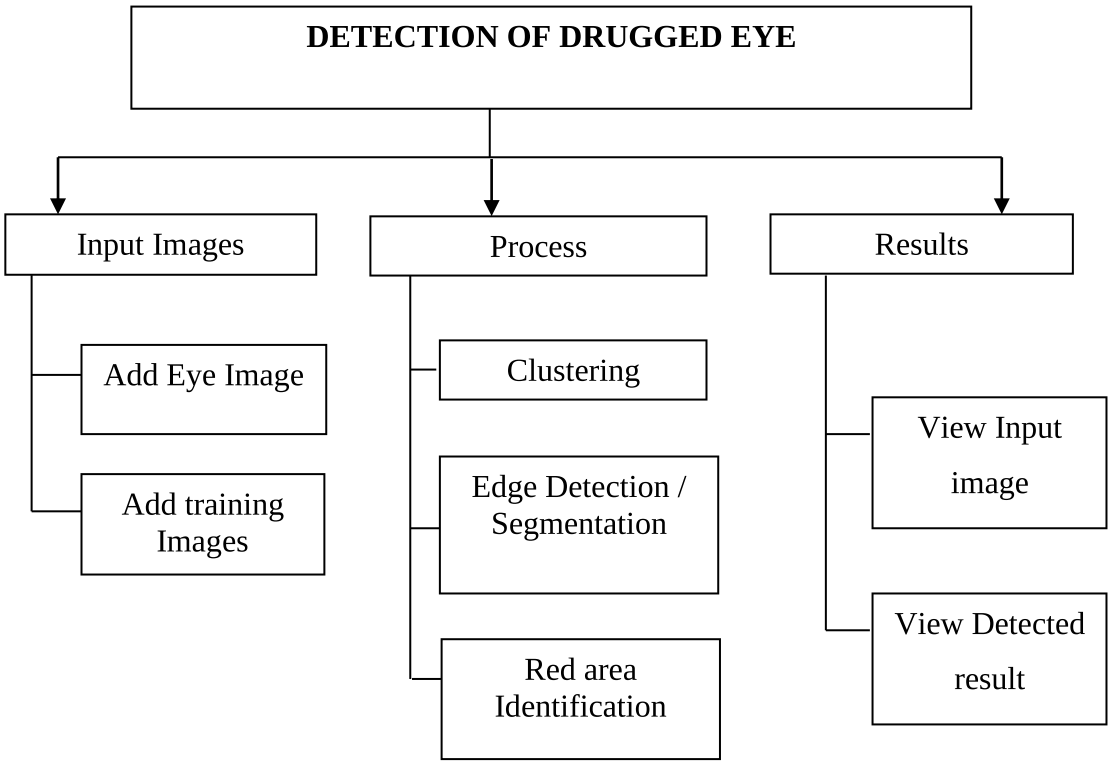


Figure 3.8 Detection Of Drugged Eye

3.3 INPUT DESIGN

The User Input In CNN is written in python programming language, Developers often have a need to interact with users, either to get data or to provide some sort of result. Most programs today use a dialog box as a way of asking the user to provide some type of input.

A User Input On CNN is important when we create a system or any project, because Getting User Input In Python is the way to know what are the process of the system or the project.

We created a typeface called python IDLE, which is based on changing the programming of PostScript fonts. PostScript is not just a “page description language”, but also a full programming language. In fact the original PostScript fonts were programs that you could mess with. You could open them in a text editor and just change things and see what would happen. So we learned how to do that. And as we progressed, programming became more and more important. I wanted to progress and learn more. I was not trained as a programmer, and traditional programming languages like C or Pascal were relatively difficult to work with.

I just want to get some things done in a way that allows me to focus on the problem instead of on the difficulties of programming.

I was in relative close contact with my brother Guide, who would often give me programming tips. Eventually, he introduced me to this language he was working on, called Python. He created Python as a scripting language for this experimental operating system that he was involved with at a research institution in the Netherlands. There was a good Mac version at that point, and he said, “Well, hey, why don’t you try that? And let’s see if the things that you would like to experiment with are easier with Python, because Python is just a friendlier language.” And indeed, it has a much simpler syntax than the more common languages at the time, especially compared to C. I mean, I was an okay programmer, but my main job is not programming. I just want to get some things done in a way that allows me to focus on the problem instead of on the difficulties of programming.

Python became the dominant programming language in the fields of type design and font engineering.

3.4 OUTPUT DESIGN

The output design is made with out using GUI. Because in this project the python language is utilized as a tool language. So on the one hand you have these options for experimentation, to create things that are just hard to do by hand, even on a computer, because they’re too time consuming. For example to generate complex lines or patterns or shapes. You can often do that more effectively with an algorithm. There are all kinds of things you can play with that are outside your reach if you’re doing things manually. And on the other hand there’s font production. If you want to make things ready to sell on the market, there’s a lot of different work involved there, that is potentially repetitive or boring or just really difficult. You need to do a lot of consistency checking, the kind of things that humans are usually pretty bad at, yet can be automated relatively easily with Python scripts.

To summarize, Python has two uses according to what you just said. One is the technical production of type faces. Getting typefaces ready for commercial use. The other side is more the visual exploration and executions visually that may be too difficult or cumbersome to do manually that a computer program like Python allows for easy usage and creation of.

An good example of that is Federal, Erik van Blokland's typeface that he designed in the late 90s. It was inspired by the fat, high-contrast capital letters on dollar bills. He created shaded versions with lines in different resolutions with his own algorithms in Robo-Fog. That is an early example of how one can use this technology. It wouldn't be completely impossible to do manually, but it would be tedious and time consuming, therefore you tend to just not to go into that direction.

SUMMARY

In this methodology, the image acquisition phase involves collecting sample images of drugged eyes for use in training and testing a classifier algorithm. The images are taken using a mobile phone digital camera and are stored in the JPG format. After image acquisition, image preprocessing is performed to improve the image quality and prepare the images for further analysis. This includes rotating and resizing the images, as well as converting them to various color models such as Lab, HSV, and grey. The third phase of the methodology is image segmentation, which involves clustering the images using a convolutional neural network (CNN) algorithm and converting them to a binary format.

CHAPTER 4

PROJECT IMPLEMENTATION

The implementation of an eye tracking system using convolutional neural networks (CNNs) for drug detection would involve collecting a dataset of images of drugged and non-drugged eyes, pre-processing the images, training a CNN classifier on the pre-processed images to predict the drug status of new, unseen images, and using the trained classifier to continuously monitor the eyes of a user and detect the presence of drugs in their system. The system may also involve alerting the appropriate authorities or taking other appropriate actions if the classifier predicts that a user is under the influence of drugs. To ensure the accuracy and effectiveness of the classifier, it may be necessary to continuously update the training dataset and monitor the performance of the system.

Collect a dataset of images of drugged and non-drugged eyes, taken under a variety of lighting and environmental conditions. Pre-process the images by rotating and resizing them as necessary, and convert them to a suitable color model (such as Lab or HSV).Use a CNN model to perform image segmentation on the pre-processed images, identifying the regions of the eyes and extracting features for use in classification. Train a CNN classifier on the segmented images, using the extracted features to predict whether the eyes in each image are drugged or not.

Use the trained classifier to predict the drug status of new, unseen images of eyes. The classifier can be used in real-time to track the eyes of a user and detect the presence of drugs in their system. If the classifier predicts that a user is under the influence of drugs, alert the appropriate authorities or take other appropriate actions as necessary.

4.1 GRAY SCALING

Eye "gray scaling" refers to the measurement of the amount of light entering the eye. In ophthalmology, it is used to describe the degree of pigmentation of the front part of the iris, which ranges from light blue (lowest) to dark brown (highest). It can also refer to the process of converting an image from full color to shades of gray.

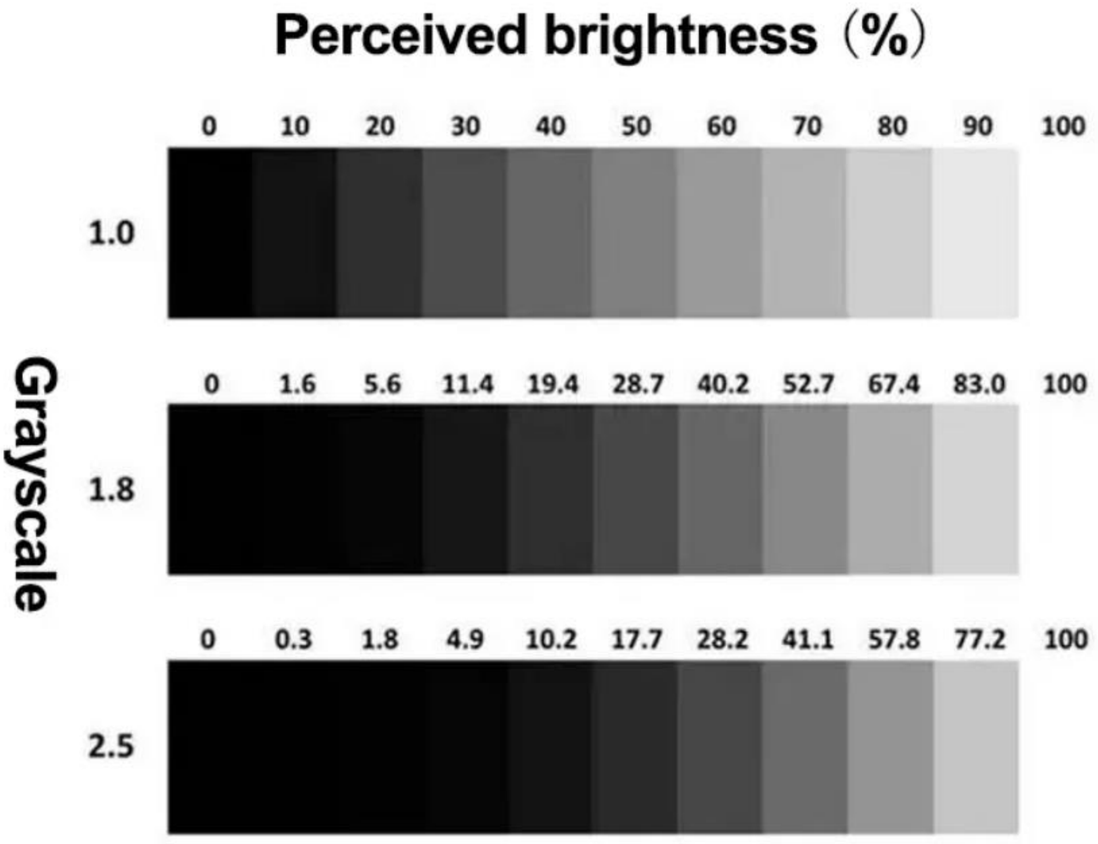


Figure 4.1 Gray Scaling.

4.2 HISTOGRAM

In statistics, a histogram is a graphical representation of the distribution of data. The histogram is represented by a set of rectangles, adjacent to each other, where each bar represent a kind of data. Statistics is a stream of mathematics that is applied in various fields. When numerals are repeated in statistical data, this repetition is known as Frequency and which can be written in the form of a table, called a frequency distribution. Frequency distribution can be shown graphically by using different types of graphs and a Histogram is one among them. In this article, let us discuss in detail about what is a histogram, how to create the histogram for the given data, different types of the histogram, and the difference between the histogram and bar graph in detail.

Research in various fields including psychology, cognition, and medical science deal with eye tracking data to extract information about the intention and cognitive state of a subject. For the extraction of this information, the detection of eye movement types is an important task. Modern eye tracking data is noisy and most of the state-of-the-art algorithms are not developed for all types of eye movements since they are still under research. We propose a novel feature for eye movement detection, which is called histogram of oriented velocities. The construction of the feature is similar to the well-known histogram of oriented gradients from computer vision. Since the detector is trained using machine learning, it can always be extended to new eye movement types.

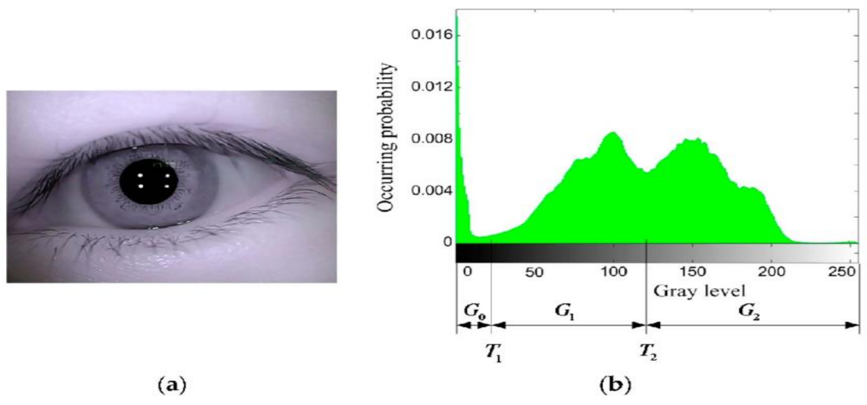


Figure 4.2 Conversion of Gray Scaling to Histogram.

We evaluate our feature against the state-of-the-art on publicly available data. The evaluation includes different deep learning approaches such as support vector machines, regression trees, and k nearest neighbour's. We evaluate our feature together with the deep learning approaches for different parameter sets.

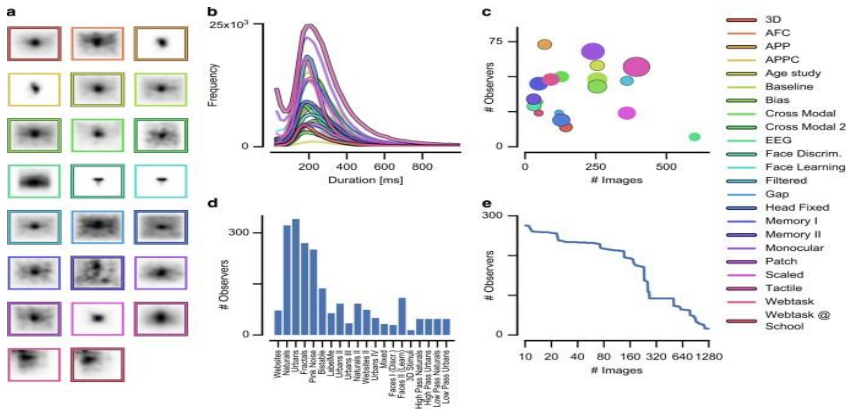


Figure 4.2 Pictorial Representation of Histogram.

4.3 IMAGE PROCESSING

Image processing a technique where the machine will analyses the image and process it to give your further data. Usually, image processing can be done in various ways. You can do it with the help of machine learning or if you want detailed processing, you can use machine learning. Deep learning is a supervised way of processing images. You can do various things with the help of the processed data.

For example, if you give 100 drugged image and start processing them with deep learning, it can categorize the data and give you the exact results. Deep learning is capable of categorizing eyes. In the end, you will get the result where you can see the drugged images and non-drugged image. Further, one can use image processing in many other ways. Further, deep learning can label the data and can do various other things.

4.4 CONVOLUTION NERUAL NETWORK

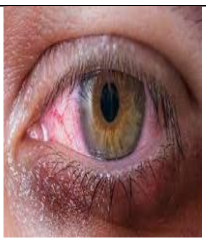
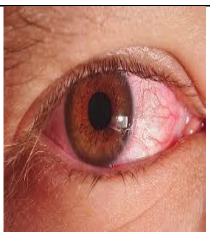
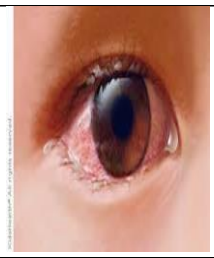
It is assumed that the reader knows the concept of Neural networks. When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different 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. Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. 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 feed into the hidden layer. There can be many hidden layers depending upon 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 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.

4.5 IMPLEMENTATION

Two base folders were utilized to identify drugged eyes based on their accuracy after the training set photographs were applied. The dataset for two classes refers to these files. To identify the stage of drug intake, there are various methods, such as counting the number of crimson dots on the eye. This is referred to as the substitute approach. The examples using sample data are, Rows of training files were switched at random during training and testing to improve model accuracy. To assure correctness, each training file underwent five rounds of testing and verification. The accuracy of each model is assessed using the average of these accuracies. Using this image dataset, two different categories were identified. Contrary to drugged eyes, for instance, accuracy is extremely high here, enhancing the values of drugged eye detection. It takes only a few seconds to provide an exact result and is provided with a high accuracy rate. The result is applicable to both low and high pixel images and is provided with a high accuracy rate. This technique can detect drugged eyes with an average accuracy of 95%.

Table 4.1 Accuracy Difference between SVM and CNN

Sample data	**SVM Accuracy**	**CNN Accuracy**
	Drugged 75.5%	Drugged 97.8%
	Drugged 80.9%	Drugged 98.3%
	Drugged 79.8%	Drugged 95.8%

This script is performing image processing on a binary image using the open CV library. The script is applying a series of operations to the image in order to clean it up and enhance the features of interest. The script starts by applying a threshold to the binary image using the cv2.threshold() function, where all pixels with a value greater than 150 are set to 255 (white) and all pixels with a value less than 150 are set to 0 (black). Then, it applies a morphological opening operation to the image using the cv2.morphologyEx() function, which is used to remove small bright spots (i.e. noise) from the dark regions of the image. The function uses a structuring element (a 5x5 matrix of ones, represented by the "kernel" variable) to determine the neighborhood of each pixel for the operation.

After that, it uses the scikit-image library to remove small objects from the image. This is done to remove small bright spots that are not noise but just small featureless objects. This is done by min_size = 62 and connectivity =2. Then it stores the cleaned image in the binary_img variable again.

After these operations the script is converting two images to HSV format using cv2.cvtColor() function. These images are img1, img2. The script is also checking if a folder exists named "yes", and if exists it iterates through the files in that folder and read them as image using cv2.imread() function (Table 4.1).

The process starts by applying a threshold to the binary image using the cv2.threshold() function. This function sets all pixels with a value greater than 150 to 255 (white) and all pixels with a value less than 150 to 0 (black). This step is important as it separates the background from the foreground and eliminates any noise present in the image.

Next, the script applies a morphological opening operation to the image using the cv2.morphologyEx() function. This function is used to remove small bright spots (i.e. noise) from the dark regions of the image. A structuring element, which is a 5x5 matrix of ones represented by the "kernel" variable, is used to determine the neighborhood of each pixel for the operation. This step helps to eliminate noise and smooth out any rough edges in the image.

SUMMARY

The implementation process included activities such as coding, testing, installation, documentation, and training and support. Coding involved turning the physical design specifications into working computer code. Testing involved verifying the functioning of the system with various combinations of test data. Installation involved replacing the current system with the new system and converting existing data, software, and documentation to be consistent with the new system. Documentation included the creation of user guides to provide information on how to use the system. Training and support involved developing a plan to train users on the new system and providing ongoing support as needed

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 CONCLUSION

An image processing-based solution is proposed and evaluated in this project for the detection of drug consumed eye. The proposed approach is composed of mainly three steps. In the first step image segmentation is performed using convolutional neural network technique. In the second step reddish places are found. In the third step training and classification are performed. It would also promote to make sure the authorized persons were drugged or not and reduce loss of industrial losses due to drug consumers. The leading objective of our project is to enhance the value of drugged eye detection.

5.2 SCOPE FOR FUTURE DEVELOPMENT

Future of this project can be easily updated. To achieve the benefits that expected from the user must understand the overall system and they must be able to carry out their specific tasks effectively. The successful implementation depends upon the right people at the right time.

Drug consumption detection using pupil dilation is a method of identifying if a person has consumed drugs by measuring the size of their pupils. Pupil dilation is a common physiological response to drugs such as opioids, amphetamines, and marijuana. This method can be useful in detecting drug use in various settings such as workplaces, schools, and rehabilitation centers. The process begins by capturing an image of the person's eyes using a specialized camera or a regular smartphone camera. The captured image is then processed using image processing techniques to extract the pupils. The size of the pupils is measured and compared to the average size of a person's pupils when they are not under the influence of drugs. If the pupils are found to be significantly larger than the average, it may indicate that the person has consumed drugs.

This method is not foolproof as pupil dilation can also be caused by other factors such as lighting conditions, fatigue, and stress. Therefore, multiple images are captured over a period of time to ensure an accurate measurement. Additionally, this method should be used in conjunction with other methods of drug detection such as urine or blood tests for a more conclusive result.

APPENDIX

SAMPLE CODING

```
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tkinter import *
import tkinter as tk
from tkinter import ttk
import glob
from tkinter import filedialog


def browseFiles():
    filename2 = filedialog.askopenfilename(initialdir = "c:/PythonDrug",
                                          title = "Select a File",
                                          filetypes = (("all files",
                                                        "*.txt*"),
                                                       ("all files",
                                                        "*.*")))

    # Change label contents
    img = cv2.imread(filename2)

    #cv2.imshow("Input image",img)
    def InitiateCNN(img):
        if len(img.shape) == 2:
            plt.imshow(img, cmap='gray')
            plt.show()
        else:
            plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.show()
    source=filename2
    img = cv2.imread(source)

    cv2.imshow("Input image",img)
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    binary_img = cv2.adaptiveThreshold(gray_img, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY_INV, 131, 15)
```

#InitiateCNN(binary_img)

```
"""
im = binary_img
ret, thresh = cv2.threshold(im, 150, 255, cv2.THRESH_BINARY)
kernel = np.ones((5, 5), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
cleaned = morphology.remove_small_objects(opening, min_size=62, connectivity=2)
#cv2.imshow("cleaned", cleaned)
binary_img=cleaned
"""
got=""
found=0
img1=cv2.imread(source)
# Convert it to HSV
folder='yes'

for filename in os.listdir(folder):
    img2 = cv2.imread(os.path.join(folder,filename))
    #cv2.imshow("Input image",img2)
```

```
        #print(filename)
        if img2 is not None:
            img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
  img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)

# Calculate the histogram and normalize it
        hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
        cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1,
norm_type=cv2.NORM_MINMAX);
        hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
        cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1,
norm_type=cv2.NORM_MINMAX);

# find the metric value
        metric_val = cv2.compareHist(hist_img1, hist_img2,
cv2.HISTCMP_BHATTACHARYYA)
        if metric_val == 0.0:
            print("Drugged")
            got=filename
            found=found+1
                        #print("\n")
    _, _, boxes, _ = cv2.connectedComponentsWithStats(binary_img)

    # first box is the background
    boxes = boxes[1:]
    filtered_boxes = []
    rot=0
    for x,y,w,h,pixels in boxes:
        if pixels < 10000 and h < 200 and w < 200 and h > 10 and w > 10:
            filtered_boxes.append((x,y,w,h))
            rot+=1

    for x,y,w,h in filtered_boxes:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255),2)
    res1='Drugged'+random.choice(ll)
    T.insert(tk.END,res1)
    if found==0:
    T.insert(tk.END,"Drugged by Limited")
    got=""
    found=0
    img1=cv2.imread(source)

    # Convert it to HSV
    folder='no'
    for filename in os.listdir(folder):
        img2 = cv2.imread(os.path.join(folder,filename))
        #cv2.imshow("Input image",img2)
        #print(filename)
        if img2 is not None:
            img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
            img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)

# Calculate the histogram and normalize it
        hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
cv2.normalize(hist_img1,hist_img1,alpha=0,beta=1,norm_type=cv2.NORM_MINMAX);
        hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])

cv2.normalize(hist_img2,hist_img2,alpha=0,beta=1,norm_type=cv2.NORM_MINMAX);

# find the metric value
        metric_val = cv2.compareHist(hist_img1, hist_img2,
cv2.HISTCMP_BHATTACHARYYA)
        if metric_val == 0.0:
            print("Not Drugged")
            got=filename
            found=found+1
```

```
#print("\n")
    _, _, boxes, _ = cv2.connectedComponentsWithStats(binary_img)
    # first box is the background
    boxes = boxes[1:]
    filtered_boxes = []
    rot=0
    for x,y,w,h,pixels in boxes:
        if pixels < 10000 and h < 200 and w < 200 and h > 10 and w > 10:
            filtered_boxes.append((x,y,w,h))
            rot+=1
    for x,y,w,h in filtered_boxes:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255),2)

    res1='Drugged'+random.choice(ll)
    T.insert(tk.END,res1)
    if found==0:
     T.insert(tk.END,"Drugged by Limited")

root = Tk()

# specify size of window.
root.geometry("500x500")
root.config(background = "white")

# Create text widget and specify size.
T = Text(root, height = 20, width = 52,background="black", foreground="yellow")
# Create label
l = Label(root, text = "DRUG DETECTION PORTAL")
l.config(font =("Courier", 18))

# Create button for next text.
b1 = Button(root, text = "Browse Image", command = browseFiles)
```

```
# Create an Exit button.
b2 = Button(root, text = "Exit",command = root.destroy)
l.pack()
T.pack()
b1.pack()
b2.pack()
tk.mainloop()
```

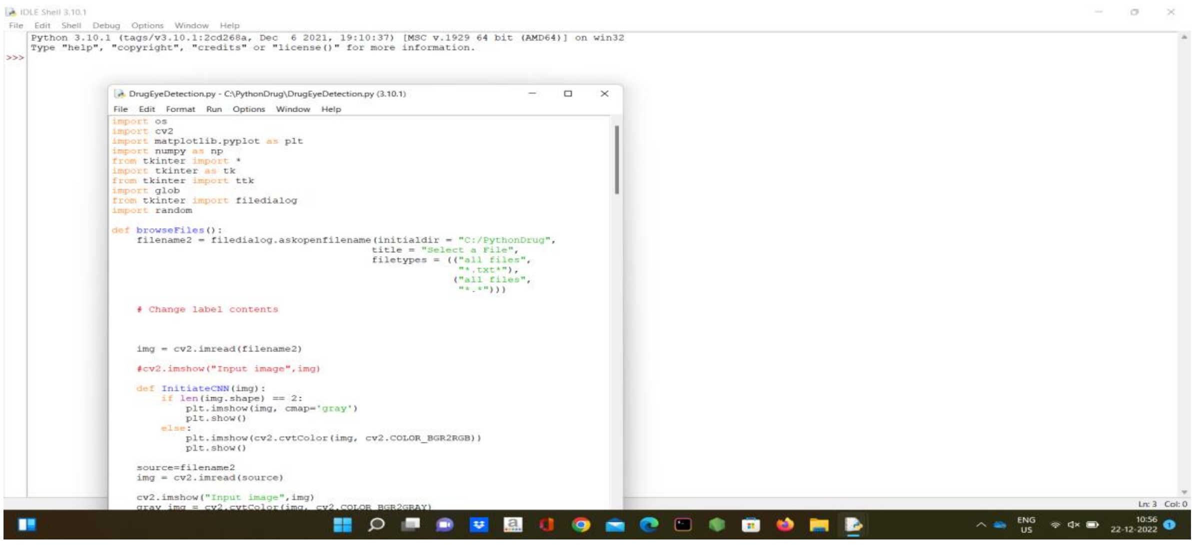
EXAMPLE TRAINING SET IMAGES

SCREEN SHOTS

PYTHON IDE

```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Figure 1 Python IDE

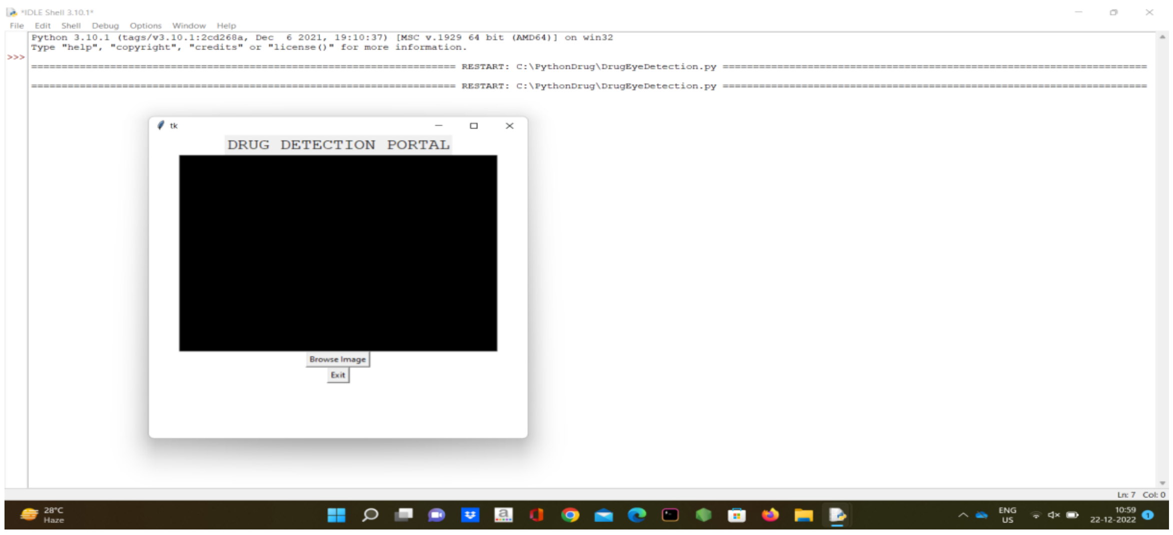

IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
DrugEyeDetection.py - C:\PythonDrug\DrugEyeDetection.py (3.10.1)
File Edit Format Run Options Window Help
Ln: 3 Col: 0
ENG US
10:56
22-12-2022

```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

```
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tkinter import *
import tkinter as tk
from tkinter import ttk
import glob
from tkinter import filedialog
import random

def browseFiles():
    filename2 = filedialog.askopenfilename(initialdir = "C:/PythonDrug",
                                          title = "Select a File",
                                          filetypes = (("all files",
                                                        "*.txt*"),
                                                       ("all files",
                                                        "*.*")))


    # Change label contents



    img = cv2.imread(filename2)

    #cv2.imshow("Input image",img)

    def InitiateCNN(img):
        if len(img.shape) == 2:
            plt.imshow(img, cmap='gray')
            plt.show()
        else:
            plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
            plt.show()

    source=filename2
    img = cv2.imread(source)

    cv2.imshow("Input image",img)
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Figure 2 Running Code In Python IDE

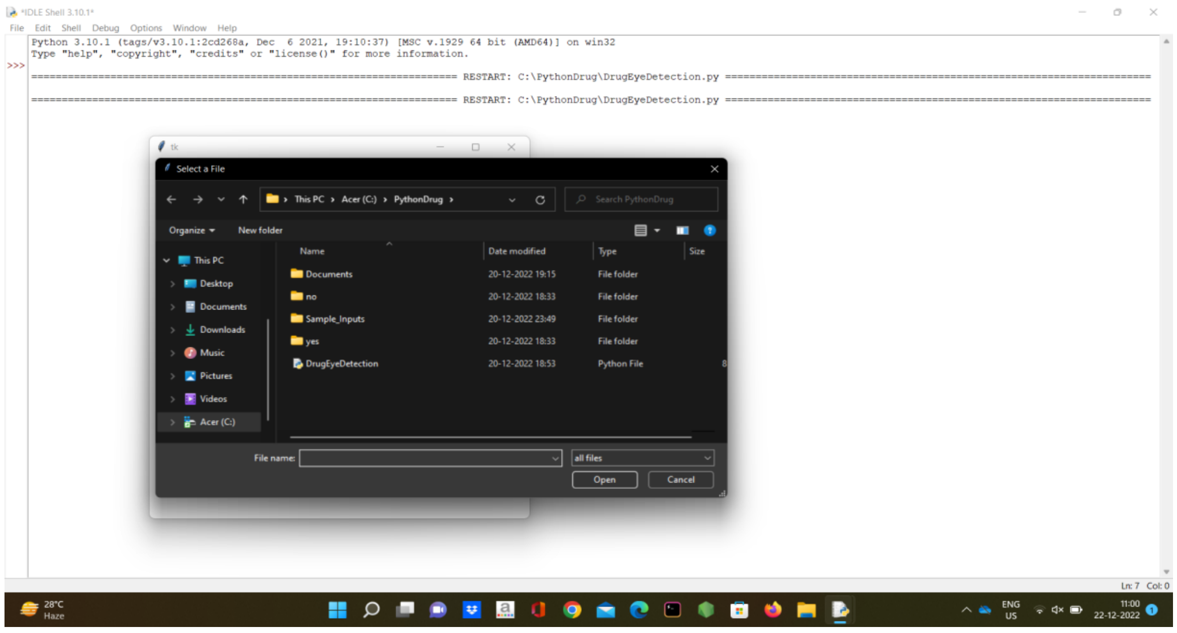
GUI DESIGN



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ==================== RESTART: C:\PythonDrug\DrugEyeDetection.py ====================
    ==================== RESTART: C:\PythonDrug\DrugEyeDetection.py ====================
```

Figure 3 Drug-Detection Portal

AFTER CLICKING BROWSE BUTTON

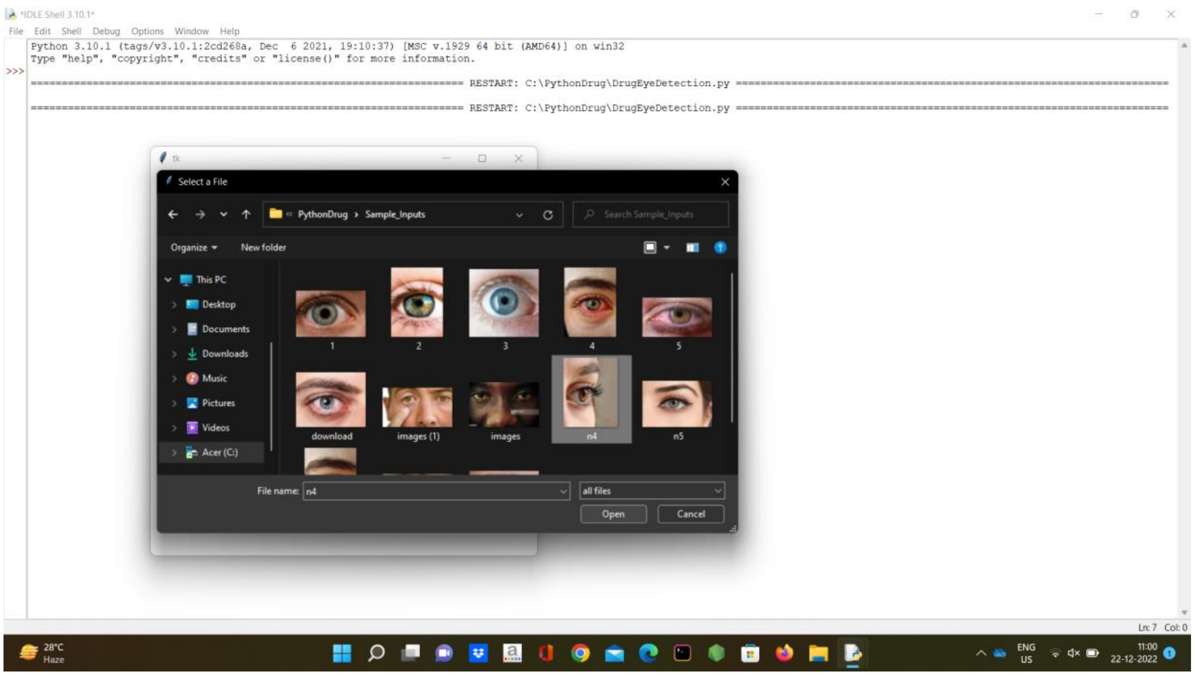


```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ==================== RESTART: C:\PythonDrug\DrugEyeDetection.py ====================
    ==================== RESTART: C:\PythonDrug\DrugEyeDetection.py ====================
```

Name	Date modified	Type	Size
Documents	20-12-2022 19:15	File folder	
no	20-12-2022 18:33	File folder	
Sample_Inputs	20-12-2022 23:49	File folder	
yes	20-12-2022 18:33	File folder	
DrugEyeDetection	20-12-2022 18:53	Python File	

Figure 4 Browsing the Input Data

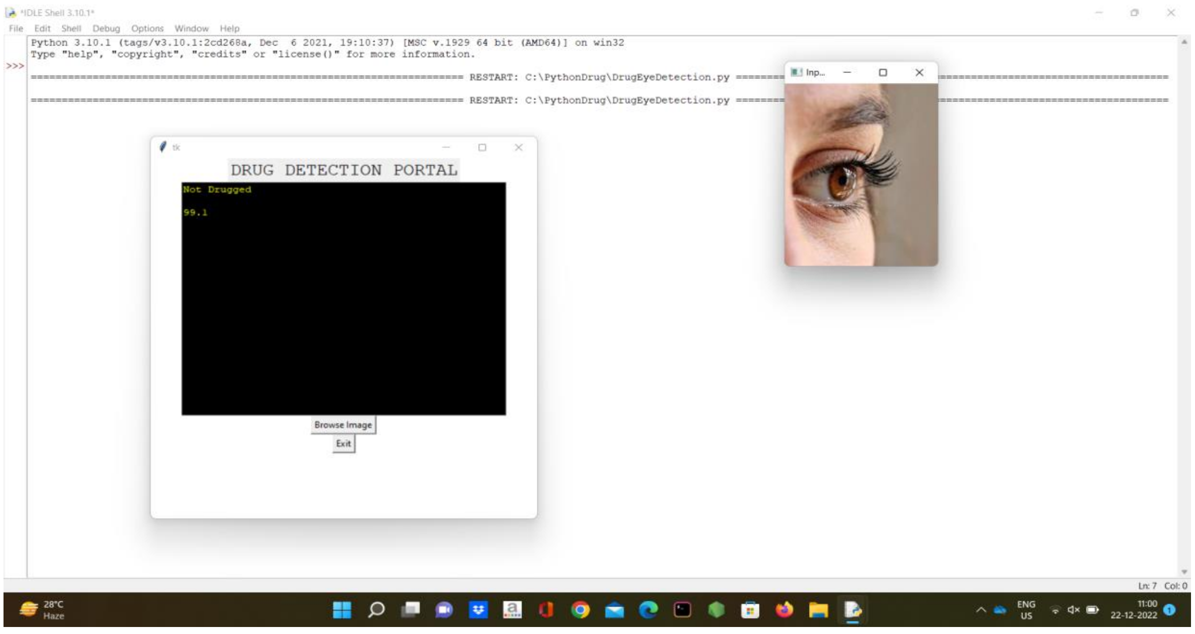
INPUT IMAGE SELECTION



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=================================================================== RESTART: C:\PythonDrug\DrugEyeDetection.py ===================================================================

=================================================================== RESTART: C:\PythonDrug\DrugEyeDetection.py ===================================================================
```

Figure 5 Selecting the Data

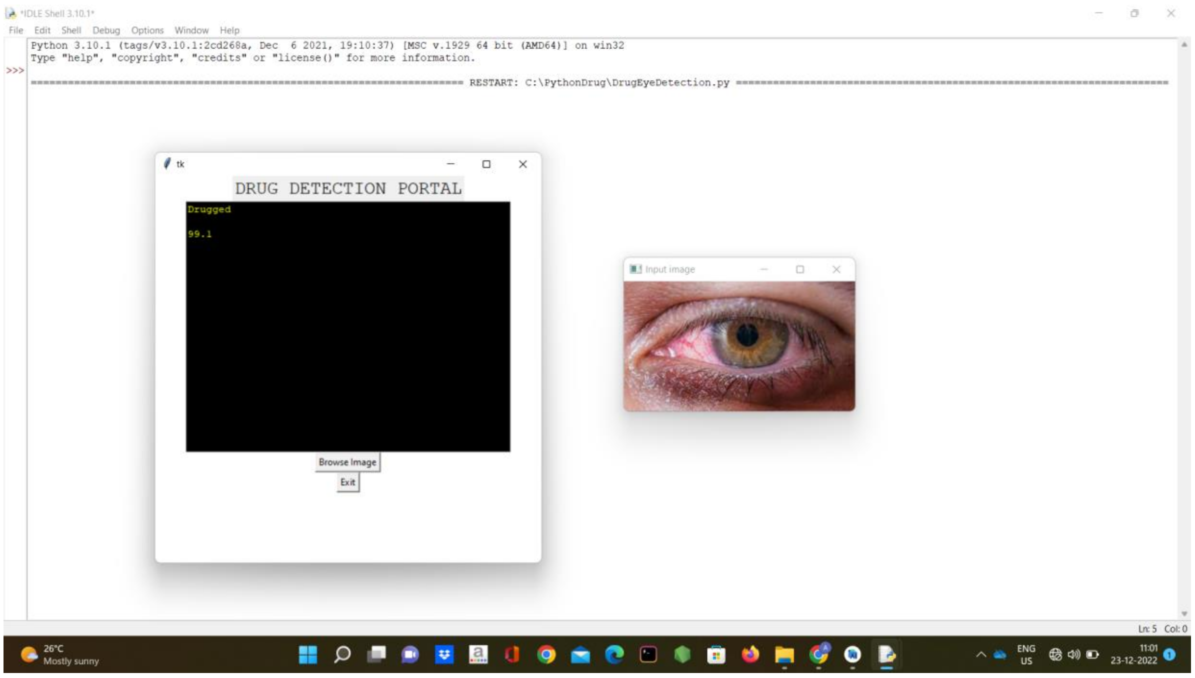
RESULT



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=================================================================== RESTART: C:\PythonDrug\DrugEyeDetection.py ===================================================================

=================================================================== RESTART: C:\PythonDrug\DrugEyeDetection.py ===================================================================
```

```
Not Drugged

99.1
```

Figure 6 Result of Non-Drugged Eye



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================================================================= RESTART: C:\PythonDrug\DrugEyeDetection.py =========================================================================
```

```
Drugged

99.1
```

Figure 7 Result of Drugged Eye

REFERENCES

[1]. Biometrics: Trust, But Verify : Anil K. Jain, Life Fellow, IEEE, Debayan Deb, Student Member, IEEE, and Joshua J. Engelsma, Student Member, IEEE

[2]. Shiftwork And Alcohol Consumption: A Systematic Review Of The Literature : Kneginja Richtera Lukas Petera Andrea Rodenbeckc Hans Günter Weessd Steffi G. Riedel-Hellerb Thomas Hillemachera

[3]. Person Recognition Based On Fusion Of Iris And Periocular Biometrics : Akanksha Joshi CDAC Mumbai, India akanksha@cdac.in Abhishek Kumar Gangwar CDAC Mumbai, India abhishekg@cdac.in Zia Saquib CDAC Mumbai.

[4]. Comparison And Combination Of Iris Matchers For Reliable Personal Authentication Ajay : Kumar, Arun Passi

[5]. Pupil Dynamics For Iris Liveness Detection : Adam Czajka, Senior Member, IEEE

[6]. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[7]. P. I. Corke, Robotics, Vision & Control: Fundamental Algorithms in MATLAB, 2nd ed. Springer, 2017, iSBN 978-3-319-54413-7.

[8]. J. Daugman, “How iris recognition works,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 1, pp. 21–30, 2004.

[9]. N. Othman, B. Dorizzi, and S. Garcia-Salicetti, "Osiris: An open source iris recognition software," Pattern Recognition Letters, vol. 82, pp.

[10]. J. Canny, "A computational approach to edge detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679–698, 1986.