IRIS TEMPLATE ATTACK DETECTION USING MACHINE LEARNING AND DEEP LEARNING METHODS

Project work submitted to Avinashilingam institute for Home Science and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

AYSHA A

20PIT003

Under the Guidance of

Dr. D. SHANMUGAPRIYA, M.Sc., M.Phil., Ph.D., SET.,

Assistant Professor and Head,

Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

COIMBATORE-641043

MAY 2022

IRIS TEMPLATE ATTACK DETECTION USING MACHINE LEARNING AND DEEP LEARNING METHODS

Project work submitted to Avinashilingam institute for Home Science and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

AYSHA A

20PIT003

Under the Guidance of

Dr. D. SHANMUGAPRIYA, M.Sc., M.Phil., Ph.D., SET.,

Assistant Professor and Head,

Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCESAND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

COIMBATORE-641043

MAY 2022

DECLARATION

DECLARATION

I hereby declare that the project entitled "**IRIS TEMPLATE ATTACK DETECTION USING MACHINE LEARNING AND DEEP LEARNING METHODS"** is a record of the original work done by **A.Aysha** (20PIT003) under the guidance of **Dr.D.Shanmugapriya, M.Sc., M.Phil., PhD., SET.,** Assistant Professor and Head, Department of Information Technology, School of Physical Sciences and Computational Sciences**,** Avinashilingam Institute for Home Science and Higher Education for Women in the partial fulfillment for the award of the degree of Master of Science in Information Technology, and this project work has not formed the basis for any Degree/Diploma/Associates.

Place:

Date:

Signature of the Candidate

Countersigned by,

Dr.D.Shanmugapriya,
M.Sc., M.Phil., PhD., SET.,
Assistant Professor and Head,
Department of Information Technology,
School of Physical Sciences and Computational Sciences**.**

CERTIFICATE

Avinashilingam Institute for Home Science and Higher Education for Women

(Deemed to be University under Estd. u/s 3 of UGC Act 1956, Category 'A' by MHRD)

Re-accredited with 'A++' Grade by NAAC. CGPA 3.65/4, Category 1 University by UGC

Coimbatore - 641 043, Tamil Nadu, India

DST - CURIE - AI Sponsored

Centre for Cyber Intelligence

CERTIFICATE OF APPRECIATION

This is to certify that Ms. Aysha A (20PIT003), M.Sc Information Technology, Avinashilingam Institute for Home Science and Higher Education for Women, has successfully completed the project entitled "Iris Template Attack Detection using ML and DL Methods" under Centre for Cyber Intelligence - Centre for Machine Learning and Intelligence - a DST - CURIE - AI facility during December 2021 - May 2022.

Dr. G. Padmavathi
Dean, School of PSCS
CCI - Principal Investigator

Dr. P. Subashini
Project Coordinator - DST - CURIE - AI

Dr. S. Kowsalya
Registrar

CERTIFICATE

CERTIFICATE

This is to certify that this project work entitled "**IRIS TEMPLATE ATTACK DETECTION USING MACHINE LEARNING AND DEEP LEARNING METHODS"** done by **A.Aysha** (20PIT003) has been submitted to Avinashilingam Institute for Home science and Higher education for women, Coimbatore-641 043 in partial fulfillment of the requirement for the award of the degree of **MASTER OF SCIENCE IN INFORMATION TECHNOLOGY.** This Project has not found the basis for the award of any Degree/Associate/fellowship or similar title to any Candidate of any University. Certified as a Bonafide record of the work submitted for the Viva voce held on _____.

Signature of the Head of the Department **Signature of the Supervisor**

Signature of the Examiner(s)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I owe my sincere thanks to **Lord Almighty** and **My lovable parents** for showering their generous blessings upon me in all endeavors.

I wish to express my gratitude to **Prof.S.P.Thyagarajan,** Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing the facilities to conduct this study.

I extend my thanks to **Dr. Bharathi Harishankar, Ph.D., FRSA.,** Vice Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing flamboyant help towards the completion of the study.

I record my deep sense of gratitude and indebtedness to **Dr. S. Kowsalya,** M.Sc., M.Phil., Ph.D., Registrar, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing adequate help for the study

I grateful record my sincere thanks to **Dr. G. Padmavathi** M.Sc., M.Phil., Ph.D., Dean and Professor, School of Physical Sciences & Computational of Sciences, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for timely help rendered throughout the course of this work.

I heartily Thank my esteemed project guide **Dr. D. Shanmugapriya,** M.Sc., M.Phil., Ph.D., SET., Assistant Professor and Head, Department of Information Technology, for imparting tremendous assistance and well-timed support for triumph of our project.

I express my honorable thanks to our project coordinator **Dr. F. Paulin,** MCA., M.Phil., Ph.D., Department of Information Technology, for imparting tremendous assistance and well-timed support for triumph of our project.

I sincerely thank all **the staff members** of Department of Information Technology,

Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for their help and support.

I would like to express my special thanks to **my parents, my friends** and all **my well-wishers** for their constant encouragement, support and help in carrying out this work successfully.

I would like to acknowledge the help rendered by Center for Cyber Intelligence, DST – CURIE – AI Sponsored Phase II for providing the laboratory facilities to execute my project.

ABSTRACT

ABSTRACT

The iris is a protected, externally visible unique organ whose epigenetic pattern remains consistent throughout adulthood. Because of these properties, it's a good candidate for use as a biometric for identifying people. Irises are unique to each person. Even identical twins and the left and right eyes of the same person have different appearances. The chances of discovering two people with identical iris patterns are estimated to be one in 1052. As biometric identification systems become more common, an attacker's incentive to stage a system compromise grows, as does the requirement to assure system security and integrity.

The objective of this project is to find the iris template attack in iris template of each user that is being stored on the background. Multiple pre-processing and classification algorithms are applied such as Eye Detection, Iris Detection, Morphological Operations, Edge Detection using Contour and Iris Segmentation. Since there is no specific template for attack, the attack, the attacked template images are created for further processing. The model is being built using deep learning technique namely Convolutional Neural Network (CNN) without max pooling which provides 97.50% Accuracy and CNN with max pooling gives 100% Accuracy. In Machine Learning (ML) techniques, logistic regression is applied to classify and detect the attacked template from genuine iris template and it gives 90% accuracy.

CONTENTS

CONTENTS

INTRODUCTION

1. INTRODUCTION

The chapter 1 consists of a brief discussion about the biometric system, its types and the general introduction to the proposed system.

1.1 WHAT IS A BIOMETRIC SYSTEM?

A biometric system allows for the systematic recognition of an individual by using some form of distinguishing feature or characteristic of human body. Fingerprints, facial traits, voice traits, geometry of hand, handwriting, the iris, and the retina, have all been used to construct a biometric system.

Biometric systems begin by obtaining the sample feature, such as a sound signal captured digitally for voice recognition or a digital captured color image for face recognition. The obtained sample is converted as a biometric template using a mathematical formula. It will generate are presentation of the characteristic that is normalized, efficient, and discriminating, which can be compared objectively to other templates to ascertain identity. Most biometric systems have two operating modes. An enrollment mode for adding templates to a database, and an identification mode for creating a template for a person and then searching the database of pre-enrolled templates for a match.

A good biometric is defined by the usage of a feature that is highly distinctive, stable, and easy to capture, with a low likelihood of two people having the same characteristic. In addition, the trait remains constant over time. To make the user's life easier and to avoid misrepresentation of the functionality.

1.1.2 THE BENEFITS OF USING A BIOMETRIC SYSTEM:

• Easier fraud detection

• Better than password/PIN or smart cards

• No need to remember passwords

• Requires actual presence of the individual to be identified

• Unique physical or behavioral trait

• Cannot be borrowed, stolen, or forgotten

1.1.3 BIOMETRICS TYPES

There are two types of Biometrics. They are described in the following figure 1.

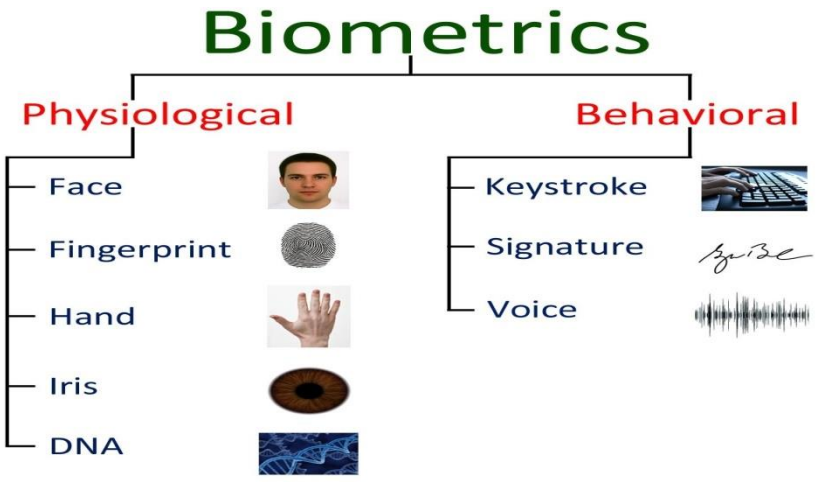


Figure 1. Types of Biometrics

1.2 IRIS

The iris is a protected, externally visible organ whose epigenetic pattern remains consistent throughout adulthood. Because of these properties, it's a good candidate for use as a biometric for identifying people. The goal of 'Iris Recognition,' a biometrics-based technique for personal identification and verification, is to recognize someone based on their iris prints. Iris patterns, in fact, are known for their consistency and uniqueness.

Irises are unique to each person. Even identical twins and the left and right eyes of the same person have different appearances. Iris patterns will not be identical in one-egged twins or future clones of a person. The iris is considered an internal organ since it is adequately protected from environmental harm by the eyelid and cornea. The most accurate and quick biometric authentication technology available today is Iris Recognition.

TABLE 1. COMPARISION OF MAJOR BIOMETRIC TECHNIQUES

Biometrics	**Universal**	**Unique**	**Permanence**	**Collectable**	**Performance**	**Acceptability**	**Potential to fraud**
Face	High	Low	Medium	High	Low	Low	Low
Fingerprint	Medium	High	High	Medium	High	Medium	Low
Iris	**High**	**High**	**High**	**Medium**	**High**	**Low**	**Low**
Signature	Low	Low	Low	High	Low	High	High
Voice	Medium	Low	Low	Medium	Low	High	High
Vein	Medium	Medium	Medium	Medium	Medium	Medium	Low
DNA	High	High	High	Low	High	Low	Low

TABLE 2. IRIS RECOGNITION SYSTEM ACCURACY

Method	Coded pattern	Misidentification Rate	Security	Applications
Iris Recognition	**Iris Pattern**	**1/1,200,000**	**High**	**High-security Facilities**
Finger printing	Fingerprints	1/1,000	Medium	Universal
Hand shape	Size, length, and thickness of hands	1/700	Low	Low-security Facilities
Facial recognition	Outline, shape and distribution of eyes and nose	1/100	Low	Low-security facilities
Signature	Shape of letters, writing order, pen pressure	1/100	Low	Low-security facilities
Voice printing	Voice characteristics	1/30	Low	Telephone Service

Tables 1 and **2** show a comparison of several biometrics approaches. The IRIS recognition system is the most stable, precise, and rapid biometric authentication technology, according to the comparison.

1.3 HUMAN IRIS

The iris is a delicate, round structured and a colored part of eye that lies between the cornea and the lens in the human eye. The pupil is a circular opening that is perforated near to the iris's centre. The iris' job is to manage the intensity of light that reaches the pupil via the sphincter and dilator muscles that control pupil size. The iris's average diameter is 12 mm, while the pupil can be anywhere between 10% - 80% of the broadness of the iris.

The epithelial layer, which includes dense pigmentation cells, is the most basic layer of the iris. Above the epithelium, the stromal layer is found. Blood veins, pigment cells, and the two iris muscles are all contained inside this layer. Iris color is determined by the density of stromal pigmentation.

The multi-layered iris has two color zones that are visible from the outside. The collarette, which has a zigzag pattern, separates the outer ciliary zone from the inner pupillary zone.

1.4 IRIS RECOGNITION SYSTEM

The goal of 'Iris Recognition,' a biometric-based technique for personal identification and verification, is to recognize someone based on their iris prints. In fact, iris patterns are known for their consistency and originality. Each person's iris is distinct. The chances of discovering two persons with identical iris patterns are estimated to be one in 1052. (Population of the earth is of the order 1010).

Iris patterns will not be identical in one egged twins or future clones of a human. Even as the individual grows older, it remains steady. Iris recognition is the most accurate and quick biometric authentication technology available.

1.5 PROBLEM STATEMENT

To identify a template attack in each user's iris template that is saved in the background.

1.6 OBJECTIVE

To provide a baseline solution for detecting iris template attacks and classifying whether they are attacks or genuine iris samples based on a variety of performance assessment parameters.

1.7 MOTIVATION AND JUSTIFICATION

It is essential to come up with a good solution to this challenge that will secure the biometric system's safety and security.

Our top aim is to detect the assault before it damages the iris biometric system and template.

To give a baseline answer to the problem, a mix of multiple pre-processing and classification algorithms are being developed and used to this suggested project.

SUMMARY

The current chapter depicts an outline of the proposed project such as its Problem Statement, Objective, Motivation and justification, as well as the composite introduction to the general biometric system, The Iris and its comparison to other biometric systems. The next chapter shows the background study done to develop this project.

REVIEW OF LITERATURE

2. REVIEW OF LITERATURE

The chapter 2 covers the literature review done towards the proposed system. The following table 3 consists of the title, techniques and the observation of the research in the literature survey.

TABLE 3. REVIEW OF LITERATURE

S.No	Author Name	Title of the Paper	Techniques Used / Algorithm Applied	Observation
1.	Richa Guptaa, and Priti Sehgala	HsIrisNet: Histogram Based Iris Recognition to Allay Replay and Template Attack Using Deep Learning Perspective	Image Pre-processing, Histogram Generation, Image Up sampling, HsIrisNet: CNN Architecture	99.39% accuracy in IITD database and 99.72% accuracy in Casia-Iris-Interval v4
2.	Christian Rathgebh , and Andreas Uhl	Statistical attack against iris-biometric fuzzy commitment schemes	Detection of pupil and iris, iris texture, Pre processed iris texture, sample iris-code.	A statistical attack based on error correction code histograms is proposed with iris-based Fussy Commitment Schemes on a comprehensive dataset.
3.	Joseph McGrath, Kevin W. Bowyer, Adam Czajka	Open Source Presentation Attack Detection Baseline for Iris Recognition	Support vector machine, Random forest and Multilayer perceptron	Exceeds 99% in NDCLD'15, and oscillates around 85% in LivDet-Iris 2017 benchmarks

4.	Vijay Kumar Sinha, Dr. Anuj Kumar Gupta and Dr. Ravinder Khanna	Detection of fake iris by using frame Difference and reflection ratio	Frame Difference Reflection Ratio	Attained 99.98% accuracy for fake iris detection method.
5.	BEDAD Fatima, ADJOUDJ Reda and BOUSAHBA Nassima	Studies of the Robustness of a Transformation-Based Multi Biometric Template Schemes Protection	LBP, 2D Log-Gabor filter, Bio Hashing, Matching.	A transformation-based protection model that focuses on Bio Hashing demonstrates how specific security and privacy may be validated
6.	R. Sujitha., N. Lalithamani	Counter Measures for Indirect Attack for Iris based Biometric Authentication. Indian Journal of Science and Technology	Lens Detection, Segmentation, Normalization, Feature Encoding, Matching	80% Accuracy

Table description:

The above table 3 provides the details of title, techniques and the observation of the research for template attack detection in image processing.

METHODOLOGY

3. METHODOLOGY

The proposed system is a model creation for template attack detection and classification. Initially, pre-processing is accomplished using eye detection, iris detection, morphological operation, edge detection and iris segmentation. Followed by template generation for both genuine template and genuine attack template is used to generate the dataset for model training and testing. The machine learning model and deep learning model, namely logistic regression and CNN is applied to classify and detect the template attack using casia iris dataset. Finally, the model is evaluated using performance metrics.

The overview of methodology is shown in figure 2

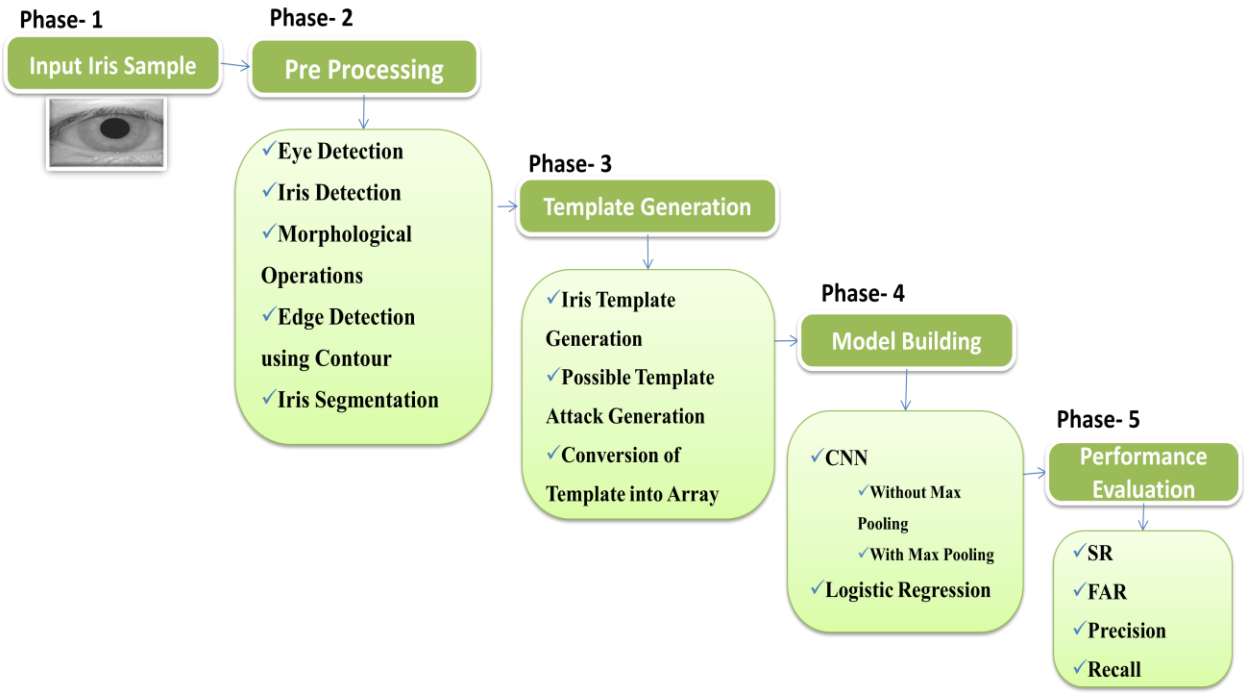


Figure 2. Methodology Overview

The entire methodology is divided into five modules, namely, image acquisition, image preprocessing, template generation, model building using machine learning and deep learning and performance evaluation to evaluate the performance.

3.1 PHASE 1: IMAGE ACQUISITION

Iris identification starts with isolating the real iris area in a digitized eye picture. The top and lower sections of the iris area are generally obscured by the eyelids and eyelashes. The imaging quality of ocular pictures determines the success of segmentation. In order to recognize the authentic user based on iris patterns, images from the CASIA iris database are utilized as input images. The preprocessing stage is crucial to generate the template and it also increase the performance of an iris pattern attack detection system, because data misrepresented as iris pattern data would damage the biometric templates generated, leads to poor user recognition.

3.2 PHASE 2: IMAGE PRE-PROCESSING

Phase 2 is segregated into five steps. Each can be explained below:

3.2.1 Eye Detection

The training of the classifier is the first phase. Casia iris pictures are required for this procedure. It is made up of positive imagery. The ocular traits will be captured in positive photos. The eyes' characteristics will then be retrieved from the photos. The Haar characteristics indicated in the diagram below are employed for detection. The approach then subtracts the total of white rectangular pixels from the sum of black rectangular pixels indicated in the picture for each feature.

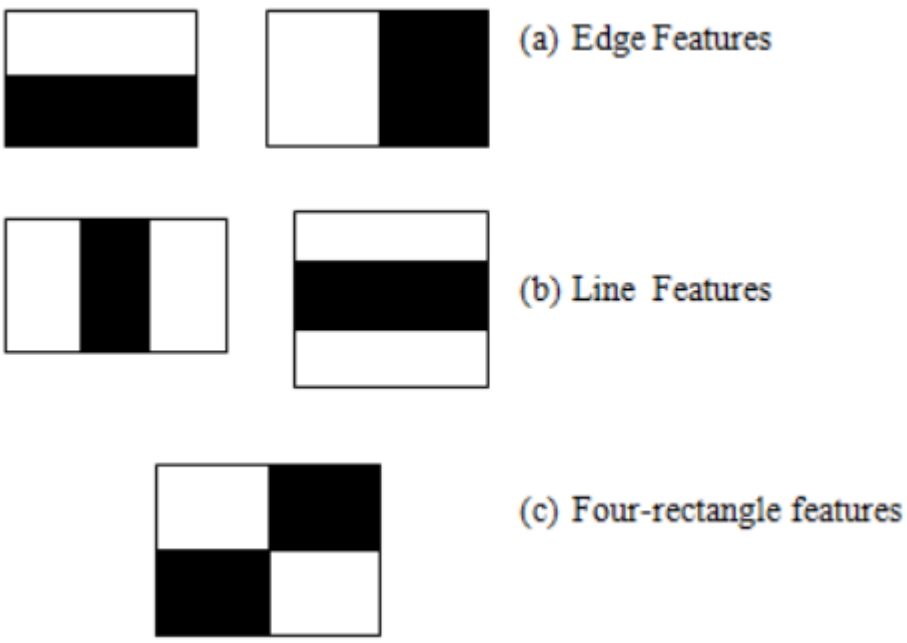


Figure 3: Haar features

The total of pixels in white and black rectangles is required for each feature computation. All of the characteristics are applied to all of the training photos in this case. It determines the appropriate threshold for each attribute that may be used to categorize photos into positive and negative categories. The primary classifier consists of a collection of weak classifiers. They're termed weak classifiers since they can't categorize the image on their own, but when used together, they can. Rather than applying all of the features to a single frame, they are clustered and applied one by one in a cascade of classifiers. If a picture fails the first set of features, it will be discarded and the other characteristics will not be examined.

OpenCV has a training approach for detecting eyeballs in images called the Haar Cascade model. You must first import OpenCV before loading the relevant XML files. These XML files provide a collection of features that will be applied to the picture. The eyes are identified using the later function detect Multi Scale, which returns a rectangle for the observed eyes.

cv2.CascadeClassifier("XML file name"): used to import Cascade classifier file (XML file). img.copy(): used to copy an image. cv2.imread('image_name'): used to extract image. plt.imshow(variable_name): used to plot the rectangle boundary for the detected eye. CascadeClassifier.detectMultiScale(): used to detect eyes.

Function used in Haar cascade classifier

3.2.2 Iris Detection

This step represents the second process where it's responsible for iris detection. The model takes the images detected in the last phase and it ensures that there is an iris inside the eyes the image will pass to the next step if it passed this step. It highlights the iris using Hough circle based on the specified iris threshold value.

3.2.3 Morphological Operations

Morphological processing is a set of non-linear processes that deal with the form or morphology of picture features. Morphological procedures, it's especially well-suited to binary image processing because it relies solely on the right and necessary ordering of image pixels rather than their numerical values. Grayscale pictures can also be subjected to morphological treatments in which the absolute data points seem to be of no or low importance, and the light output signals are unknown.

To study a picture, morphological techniques use a little form or pattern known as a structuring or structured elements. The structured element is placed in all available positions in the picture and compared to the pixels in its immediate vicinity. Some operations determine if an element "fits" into the neighborhood, while others determine whether it "reaches" or overlaps it.

Most morphological procedures, such as the complementary of a binary picture, are expressed as a mixture of eroding, dilatation, and simple set operations:

$$f^c(x,y) = 1 \text{ if } f(x,y) = 0, \text{ and } f^c(x,y) = 0 \text{ if } f(x,y) = 1,$$

the **intersection** $h = f \cap g$ of two binary images f and g:

$$h(x,y) = 1 \text{ if } f(x,y) = 1 \text{ and } g(x,y) = 1, \text{ and } h(x,y) = 0 \text{ otherwise,}$$

and the **union** $h = f \cup g$ of two binary images f and g:

$$h(x,y) = 1 \text{ if } f(x,y) = 1 \text{ or } g(x,y) = 1, \text{ and } h(x,y) = 0 \text{ otherwise:[10]}$$

3.2.4 Edge and Contour Detection

Edges appear at locations in a picture where the brightness values vary greatly, and as a result, they frequently show the edges, or occluding limits, of the objects in the scene. Large luminance shifts, on the other hand, might correlate to surface marks on objects. Tangent discontinuities in the brightness signal might potentially indicate a scene object border.

As a result, the first challenge in modelling this biological process is accurately defining what an edge is. Defining edges as image signal step gaps or discontinuities is the most popular way. Finding possible values in the signal's derivative or zero-crossings within the signal's second derivative becomes a common approach of localizing these discontinuities.

Edge detection in computer vision has historically been accomplished by convolving the signal with a linear filter that approximates a first or second derivative operator. A first derivative will be approximated by an odd symmetric filter, and peaks in the convolution output will correspond to picture edges (luminance discontinuities).

A second derivative operator will be approximated by an even symmetrical filter. Edges will be represented by zero-crossings within the result of convolution with the even symmetrical filter, while maxima will be represented by parallel discontinuity, sometimes known as lines or bars.

Contour detection isn't the only image segmentation technique; there are a slew of others, including semantic segmentation, the Hough transform, and K-Means segmentation, to name a few.

Algorithmic steps of contour edge detection are explained below:

- Converting the image into a binary image; it is usual to use a binary image as the input image (which it has to be the outcome of image thresholding or background subtraction).
- findContours() is an OpenCV method used for detecting contours.
- Show the output with these outlines.

3.2.5 Iris Segmentation using Hough Transform

The fundamental feature extraction approach for recognising circles in defective pictures in digital image processing is circle Hough Transform (CHT). The circle candidates are created by picking local maxima in an accumulator matrix and then "voting" in the Hough parameter space.

Algorithmic steps of Hough Transform using opencv is explained below:

1. Initialize the accumulator (H[a,b,r]) to all zeros.
2. Find the edge image using any edge detector.
3. For r= 0 to diagonal image length. For each edge pixel (x,y) in the image. For $\Theta = 0$ to 360.

$$a = x - r*\cos\Theta \quad b = y - r*\sin\Theta \ ...$$

4. Find the [a,b,r] value(s), where H[a,b,r] is above a suitable threshold value.

3.3 PHASE 3: TEMPLATE GENERATION

A template-based method may be useful for templates with few characteristics or when the majority of the template picture serves as the matching image. As previously stated, because while template-based matching may necessitate a high amount of sampling points, the sample size can be reduced by lowering the quality of the searching and pattern images by the same factor and doing sample. Offering a search frame of data points inside the search picture so the template does not have to search every potential data point, and providing a search window of data points within the search image. Such that the template does not have to seek every feasible piece of data, or a mix of both.

So, the template is generated for genuine iris image gathered from the above mentioned pre-processing techniques and stored it as genuine iris template. The possible attack in done in some samples of generate iris template and stored it as attacked iris template.

3.3.1 Converting Template Image into an Array Format

In this step, it is used to convert an image to NumPy array and then save that array into CSV file. It is accomplished by applying PIL and numpy library. The two csv file is generated for both the genuine iris template and attacked iris template.

3.4 PHASE 4: MODEL BUILDING

In this phase, the model building is done using the csv file gathered from the previous phase. In this phase, the csv file is trained using machine learning model and deep learning model to classify and detect the iris template attack.

3.4.1 Machine Learning:

Models of machine learning are strong tools that may be utilised to accomplish critical jobs and solve complicated issues quickly and effectively. These models are prepared for usage in a variety of industries due to the exponential growth of data in the modern world. Machine learning in finance can be utilized to proactively monitor bank transfers for signs of fraud, and machine learning in healthcare can be applied to power the next generation of diagnostic tools.

The process of creating a ML model is typically complicated, and it is guided by data science experts. However, as more businesses employ machine learning, a thorough grasp of the process is essential.

Algorithm steps for machine learning techniques:

- Contextualize machine learning in your organisation
- Explore the data and select the appropriate algorithm
- Prepare and clean the dataset
- The prepared dataset is splits into train and test data, it is used to perform cross validation
- machine learning model is optimized
- The model is deployed

There are three types of machine learning algorithms: supervised learning, unsupervised learning, and reinforcement learning.

1. **Supervised Learning -** When data is labeled and the goal is to categorise or predict a value, this is referred to as supervised learning. Detecting fraudulent transactions in a list of credit card transactions, for example.

2. **Unsupervised Learning** - When data is not labeled and the goal is to detect patterns in the data, this is known as unsupervised learning. In this situation, you're instructing the machine learning model to analyze the data and derive conclusions from it. Customer segmentation based on spend data, for example.

3. **Reinforcement Learning** - Trial and error learning. This is the most human-like method of learning. The goal is to discover the best policy for acting in a particular situation. The machine learning model considers all options, develops a policy that optimises benefit, and then executes that policy (trial). Apply reinforcements back into the process if the first policy has flaws, and keep doing so until you find the optimal policy.

Because supervised learning is used to categorise or forecast a value, there are two types of supervised learning algorithms: classification and regression models.

1. **Classification model** - A classification model, in basic terms, predicts alternative outcomes. Predicting whether or not a transaction is fraudulent is an example.

2. **Regression model** - This type of model is used to forecast a numerical value. Predicting the sale price of a property, for example.

Logistic Regression

Logistic regression is a good example of supervised learning. It's used to calculate or forecast the likelihood of a binary (yes/no) event occurring. Logistic regression is used to determine whether or not a person is likely to be infected with COVID-19.This is known as binary classification as there are only two potential answers to this question: yes, they are infected, or no, they are not infected.

In this hypothetical scenario, the possibility of a person getting infected with COVID-19 might be determined by the viral load, symptoms, and the existence of antibodies, among other

factors. Our factors (independent variables) would include viral load, symptoms, and antibodies, all of which would affect our result (Dependent Variable).

The sigmoid function is a mathematical formula that converts expected values into probabilities. It converts any real number from 0 to 1 into another number. The logistic regression's value must be either 0 or 1, and it cannot exceed this limit, resulting in a "S"-shaped curve. The Logistic function or sigmoid function is the name for the S-form curve. The threshold value, which indicates the likelihood of either 0 or 1, is used in logistic regression. For example, numbers above the threshold value likely to be 1, whereas values below the threshold value tend to be 0.

Logistic regression is generally a good fit for training data that meets the following assumptions.

- The expected result is either binary or dichotomous.
- The elements that impact the result, or independent variables, are unrelated to one another. In other words, there is little or no inter correlation in the independent variables.
- The data points and the log chances can be linearly connected.
- The sample sizes are quite huge.

If the training data doesn't meet the aforementioned criteria, the logistic regression may not be suitable for that particular application.

Algorithm steps of logistic regression

- Pre-processing of data
- Training the given Set and Logistic Regression
- Predicting the outcome of a test
- Check the result's accuracy (Confusion matrix creation)
- Creating a visual representation of the real test results.

3.4.2 Deep Learning:

Machine Learning, on the other hand, is a subset of Artificial Intelligence, while Deep Learning is a subset of Machine Learning. Artificial intelligence (AI) is a broad phrase that refers to methods that allow computers to emulate human behaviour. All of this is made possible through machine learning, which is a set of algorithms taught on data. Deep Learning, on the other hand, is a sort of Machine Learning inspired by the human brain's structure. By continuously evaluating data with a particular logical framework, deep learning computers aim to derive similar conclusions as humans would. Deep learning does this through the use of neural networks, a multi-layered structure of algorithms.

The architecture of neural network is modelled after the human brain's organisation. Neural networks can be trained to perform the same tasks on data as our brains do whenever it comes to finding patterns and categorising various types of data. Individual layers of neural networks can be viewed as a type of filter that works from the most obvious to the most subtle information, increasing the chances of recognising and producing the correct output. In a similar way, the human brain functions. Our brain strives to compare new information to previously experienced items as we acquire it. The same principle applies to deep neural networks.

The lack of requirement for so-called feature extraction is the primary benefit of deep learning over machine learning. Traditional machine learning approaches were widely utilised long before deep learning. Decision Trees, SVM, Nave Bayes Classifier, and Logistic Regression are some examples. Flat algorithms are another name for these algorithms. The term "flat" refers to the fact that these algorithms are typically unable to be applied directly to raw data (such as .csv, images, text, etc.). A pre-processing procedure called Feature Extraction is required.

Feature extraction is generally fairly difficult and necessitates a thorough understanding of the issue domain. For best results, this pre-processing layer must be customised, tested, and optimised across numerous rounds. Deep Learning's artificial neural networks, on the other hand, are artificial neural networks. The Feature Extraction phase is not required for them.

The biological neurons in our brains serve as inspiration for artificial neural networks. In truth, artificial neural networks mimic some of the core functions of human brain's neural networks, although in a reduced form. Let's start with biological neural networks and see how they compare to artificial neural networks. In a nutshell, a biological neural network is made up of many neurons.

A series of interconnected units or nodes makes up a neural network. These nodes are referred to as neurons. These artificial neurons are based on our brain's organic neurons. Simply said, a neuron is a graphical depiction of a numerical value (e.g. 1.2, 5.0, 42.0, 0.25, etc.). In a true biological brain, each link between two artificial neurons can be called an axon. Weights, which are also nothing more than numerical numbers, are used to create connections between neurons.

CNN

A Convolutional Neural Network (CNN) is kind of neural network that specialises in processing data with a grid-like architecture, such as an image. A digital picture is a representation of visual data in binary form. It consists of a grid-like arrangement of pixels with pixel values indicating how bright and what colour each pixel should be. The second we perceive a picture, our brain analyses a massive quantity of data. Each neuron has its own responsive field and is coupled to other neurons in such a way that the full visual field is covered. In the biological vision system, each neuron can only respond to stimuli in a small area called the receptive field, and each neuron in a CNN analyses data only in its receptive field. The layers are designed such that simpler patterns (lines, curves, etc.) are detected first, followed by more complicated patterns (faces, objects, etc.). One may give computers sight by utilizing a CNN.

A convolutional layer, a fully connected layer, and a pooling layer are the three layers that make up a CNN

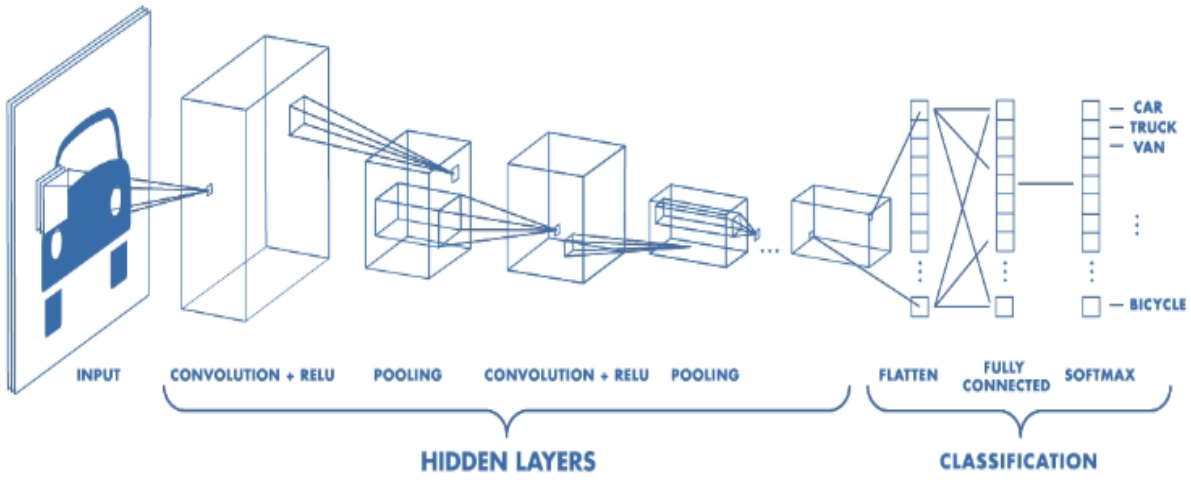


Figure 4: Basic Architecture of CNN

Convolution Layer

The CNN's main building piece is the convolution layer. It is responsible for the majority of the network's computational burden. This layer performs a linear combination between two matrices, one of which is a set of trainable parameters known as a kernel and the other is the responsive field's limited section. Although the kernel is smaller than a photograph, it has more depth. If the image has three RGB channels, the kernel height and width will be modest, but the depth will span all three.

Pooling layer

The pooling layer uses a summary statistic of neighbouring outputs to substitute the network's output at certain spots. This reduces the representation's spatial size, which reduces the

amount of computation and weights necessary. Every slice of the representation is independently handled throughout the pooling procedure.

Fully Connected Layer

As in ordinary FCNN, neurons in this layer have complete connection with all neurons in the preceding and following layers. This is why a matrix multiplication followed by a bias effect may be used to compute it. The FC layer aids in the mapping of representations between input and output.

Non-Linearity Layers

Because convolution is a linear operation and pictures are far from linear, non-linearity layers are typically included right after the convolutional layer to introduce non-linearity to the activation map.

Non-linear operations come in a variety of forms, the most common of which are:

1. Sigmoid

The mathematical version of sigmoid non-linearity is () = 1/(1+e). A real-valued number is "squshed" into a range of values from 0 to 1. The gradient becomes virtually negative when the activation occurs at either tail, which is a very unfavourable feature of the sigmoid. If the local gradient becomes very small, back propagation effectively "kills" the gradient. The sigmoid's output will be either all positives or all negatives if the data entering the neuron is always positive, producing a criss dynamical weight gradient changes.

2. Tanh

Tanh reduces the range of a real-valued number to [-1, 1]. The activation saturates similarly to sigmoid neurons, however unlike sigmoid neurons, the output is zero centred.

3. ReLU

In recent years, the Rectified Linear Unit (ReLU) has grown in popularity. The function ()=max (0,) is computed. To put it another way, the activation is simply set to zero. ReLU is more dependable than sigmoid and tanh, and it speeds up convergence by six times.

Algorithm steps of CNN

Step 1: Select a Dataset.
Step 2: Prepare the Training Dataset.
Step 3: Gather data for training.
Step 4: Rearrange the Dataset.
Assigning Labels and Features (Step 5)
Step 6: Converting labels to categorical data and normalising X.
Step 7: Separate X and Y for CNN.
Step 8: Create the CNN Model by defining, compiling, and training it.
Step 9: Model Accuracy and Score

3.5 PHASE 5: PERFORMANCE EVALUATION

The quality of a statistical or machine learning model is measured using evaluation metrics. Any project requires evaluating machine learning models or methods. There are a variety of evaluation measures that may be used to test a model. In terms of a binary classification issue,

The classification here is whether the image is genuine template or attacked template. The following are some popular words to be aware of:

- True positives (TP) are predictions that turn out to be true.
- False positives (FP) are when a positive outcome is predicted but the outcome is really negative.
- True negatives (TN) are predicted negatives that turn out to be true.
- False negatives (FN) are predictions that turn out to be true.

Accuracy

The most often used statistic for evaluating a model, however it is not a reliable predictor of its performance. When courses are unbalanced, the situation becomes even worse.

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Consider the case of template attack detection. The odds of developing template attacks are quite slim. Let's assume that out of 100 templates, 90 do not have template attack and the remaining ten have. We don't want to overlook a template that has template attack yet is going unnoticed (false negative). The accuracy of detecting every template that not having any attack is 90 percent. In this case, the model did nothing but forecast that all 100 predictions would be genuine iris template.

True Positive Rate/Recall/Sensitivity

Positive incidences as a percentage of the overall number of positive incidents. As a result, the numerator (TP + FN) represents the total number of positive cases in the collection.

Assume you're trying to figure out 'how many more correct ones the model missed when it presented the right ones'.

$$\frac{TP}{TP + FN}$$

Specificity

Negative incidences as a percentage of the overall number of negative incidents. As a result, the numerator (TN + FP) represents the total number of negative instances in the collection. It's comparable to recollection, except the emphasis is on bad events. For instance, how many healthy people were told they didn't have cancer when they didn't. It's a kind of test to check how distinct the classes are.

$$\frac{TN}{TN + FP}$$

F1 score

Precision and recall have a harmonic mean. This factor considers both, therefore the greater the F1 score, the better. As you can see, if one goes low in the numerator, the ultimate F1 score drops down dramatically. So, if the positives predicted are indeed positives (accuracy) and the model does not miss out on positives and forecasts them negative, it does well in the F1 score (recall).

One disadvantage is that accuracy and recall are given equally applied, which means that depending on our application, one may be more important than the other, and F1 score may not be the best metric for it. As a result, using a weighted-F1 score or looking at the ROC or PR curve can be beneficial.

Loss:

To reduce algorithmic error, neural networks employ optimization algorithms such as stochastic gradient descent. A Loss Function is used to calculate this mistake. It's used to determine how well or poorly the model is doing.

Using the above-mentioned performance metrics, the applied deep learning model and machine learning model is evaluated to compare its performance and conclude the best from it.

SUMMARY

The Chapter 3 explained about the research methodology handles to develop a solution that detects the template attack in the iris images. The results and discussions are being described in the following sessions.

RESULTS AND DISCUSSIONS

4. RESULTS AND DISCUSSIONS

This chapter covers the results and discussions gained during the development of the suggested methodology

PHASE 1: IMAGE ACQUISITION

The CASIA IRIS dataset is gathered from kaggle repository. Figure 4 explains the iris dataset that contains the iris information of each user. It consists of 756 iris images from 108 Individuals grouped into 108 folders.

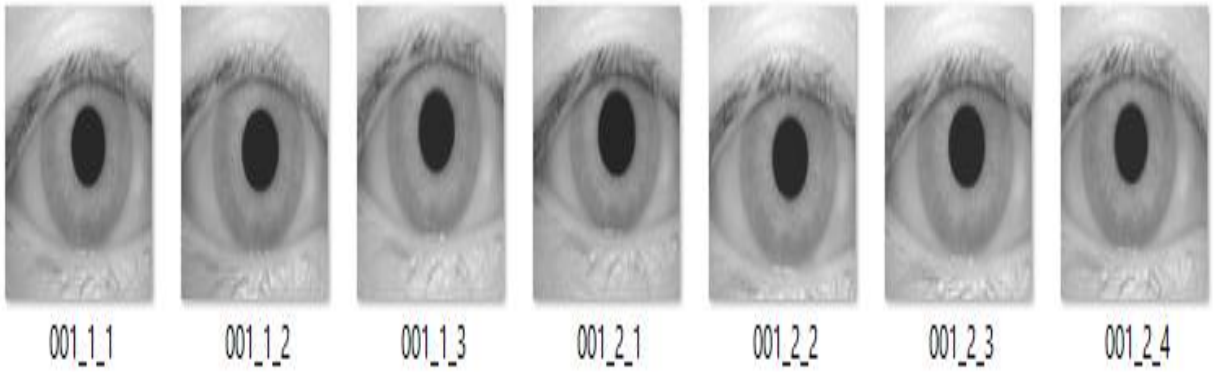


Figure 5: Sample of CASIA iris dataset

The above figure 5 shows that sample dataset images that are used in pre-processing phase. This dataset undergoes preprocessing techniques in next phase.

PHASE 2: IMAGE PREPROCESSING

- **Eye Detection**

The dataset is used to perform three image pre-processing techniques namely eye detection, iris detection, morphological operation, edge detection and iris segmentation. Figure 6 shows the outcome of cascade classifier that is used to detect the eyes using CASIA iris dataset.

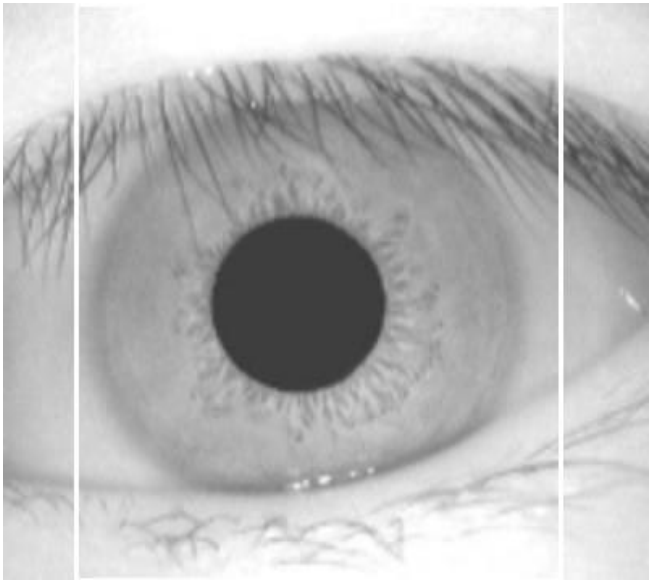


Figure 6: Eye Detection using Cascade Classifier

Description:

The detected eyes are further used to detect and highlight the iris using Hough circle. The iris is detected based on the threshold value.

- **Iris Detection**

Figure 7 shows that the specific iris part is detected using Hough circles based on the threshold value.

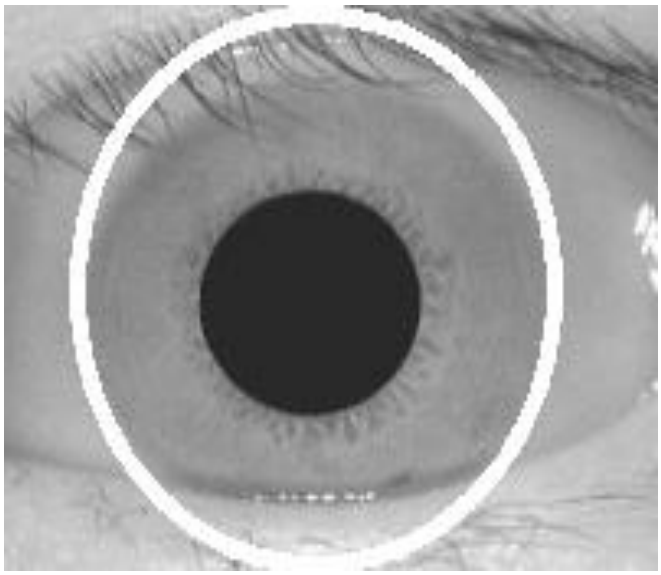


Figure 7: Iris detection using Hough circles

Description:

This iris part is detected for all the images in iris dataset. i.e., the total number of irises detected is 756 from the total number of 756 images.

- **Morphological Operations**

Figure 8 explains the morphological operations such as erosion and dilation of iris detected gray scale image.

(a) Erosion (b) Dilation

Figure 8: Morphological operations in iris image

Description:

The morphological operation is used to check whether the detected iris contains the iris information or not. If the iris information is available, then the edge will be detected in next pre-processing steps.

- **Contour based Edge Detection**

Figure 9 explains the contour based edge detection using gray scale image

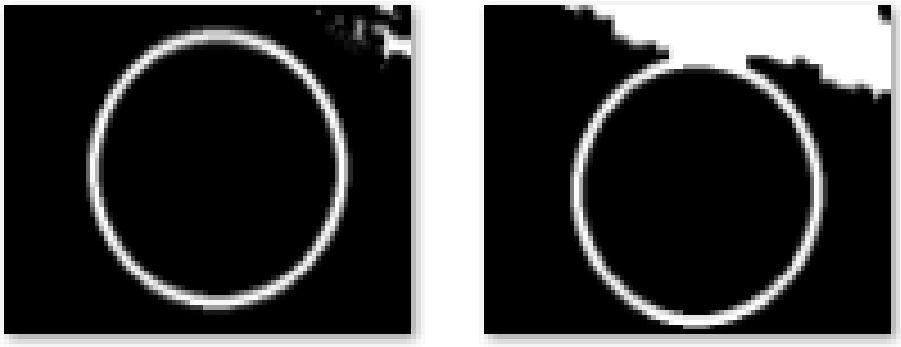


Figure 9: Contour based Edge Detection in Gray Scale Image

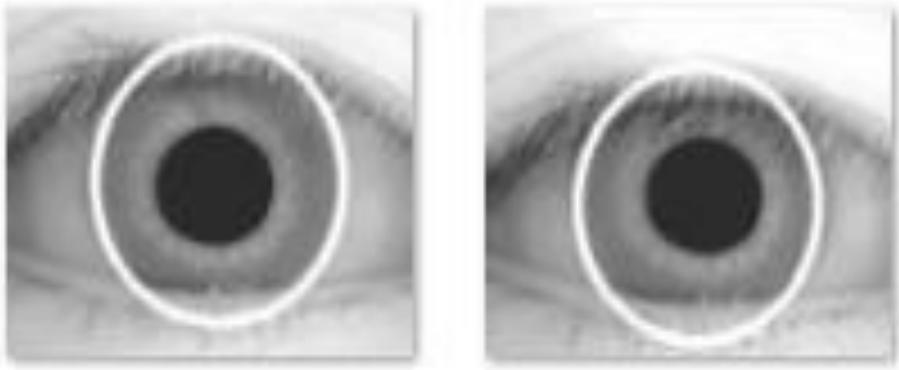


Figure 10: Outcome of Contour based Edge Detection

Description:

From the Figure 9, it is observed that the contour edge is detected using Gray scale image where the iris is highlighted using Hough circle. The Gray scale image is converted into the original image. The edge of iris is perfectly detected using contour edge detection in figure 10.

- **Iris Segmentation using Hough Transformation**

Figure 11 displays the segmented part of the iris using Hough transform.

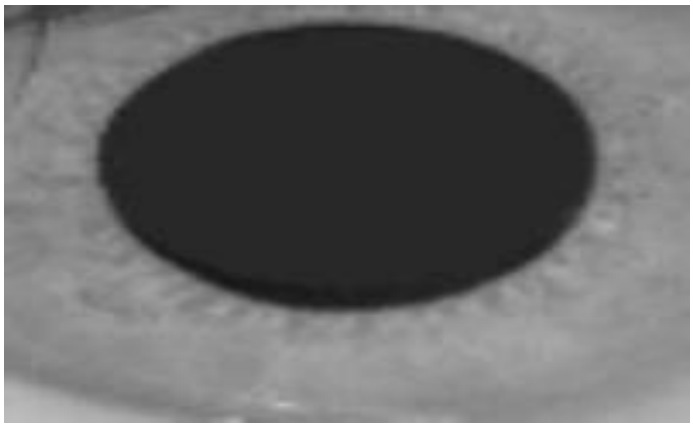


Figure 11: Iris segmentation using Hough Transformation

Description:

The detected iris edge performs segmentation using Hough transform to extract the features of iris. From the figure 11, it is observed that detected iris part is segmented using Hough transform. Hough transform is used to extract the valuable information for object detection. i.e., iris is the valuable information for iris segmentation.

PHASE 3: TEMPLATE GENERATION

The pre-processed image of user eye is now considered as genuine template. Now, the attack is performed in some genuine template by modifying the image. It is considered as attacked template.

Figure 12 shows the attacked template from the genuine template image.

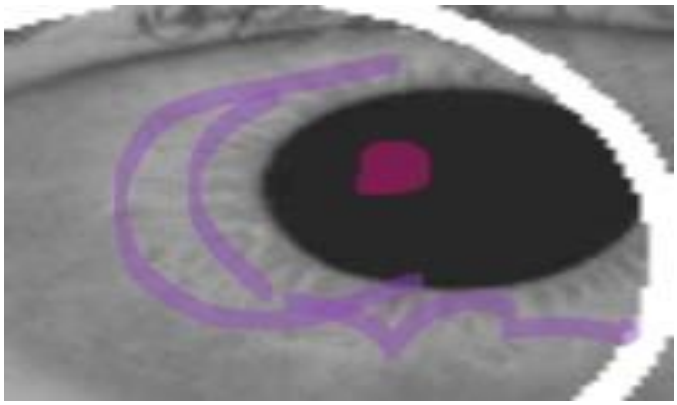


Figure 12: Attacked template from the genuine template image.

Description:

The genuine template is further generated into an attacked template as shown in the above figure 12.

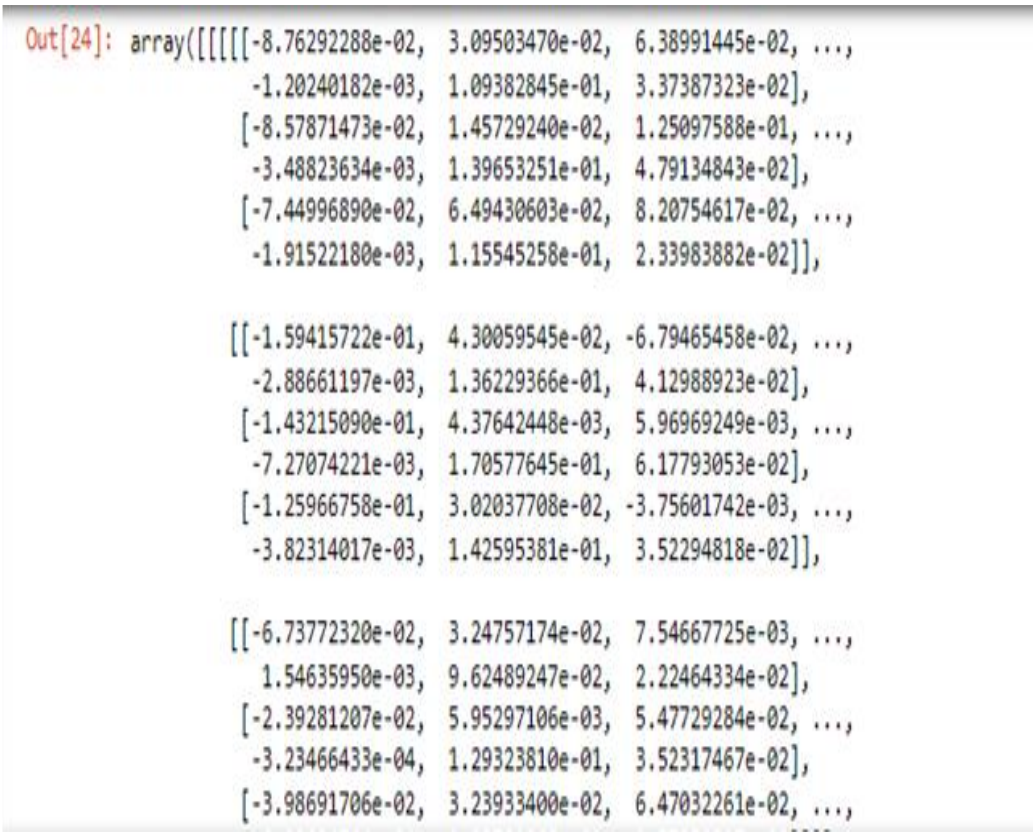


Figure 13: Both template image into array format

Description:

Both the genuine template and attacked template is now converted into array format as shown in figure 13 and store it as csv for further model building.

- **Template Information in Table form:**

Figure 14 displays the template information in table format.

	0	1	2	3	4	5	6	7	8	9	...	67490	67491	67492	67493	67494	67495	6
0	0.208497	0.208497	0.208497	0.219608	0.219608	0.219608	0.240523	0.240523	0.240523	0.246405	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0
1	0.431373	0.431373	0.431373	0.433333	0.433333	0.433333	0.445098	0.445098	0.445098	0.460784	...	0.681699	0.686928	0.686928	0.686928	0.694118	0.694118	0.6
2	0.888235	0.888235	0.888235	0.870588	0.870588	0.870588	0.846405	0.846405	0.846405	0.825490	...	0.719608	0.704575	0.704575	0.704575	0.758824	0.758824	0.7
3	0.984314	0.984314	0.984314	0.776471	0.776471	0.776471	0.379085	0.379085	0.379085	0.260784	...	0.231373	0.231373	0.231373	0.231373	0.233333	0.233333	0.2
4	0.725490	0.725490	0.725490	0.721569	0.721569	0.721569	0.701307	0.701307	0.701307	0.647712	...	0.635948	0.635294	0.635294	0.635294	0.635294	0.635294	0.6
5	0.680392	0.680392	0.680392	0.774510	0.774510	0.774510	0.943791	0.943791	0.943791	0.992810	...	0.169935	0.169281	0.169281	0.169281	0.172549	0.172549	0.1
6	0.380392	0.380392	0.380392	0.380392	0.380392	0.380392	0.383660	0.383660	0.383660	0.389542	...	0.292157	0.306536	0.306536	0.306536	0.311765	0.311765	0.3
7	0.996078	0.996078	0.996078	0.994118	0.994118	0.994118	0.995425	0.995425	0.995425	0.996732	...	0.394771	0.394771	0.394771	0.394771	0.382353	0.382353	0.3
8	0.348366	0.348366	0.348366	0.341176	0.341176	0.341176	0.330719	0.330719	0.330719	0.324837	...	0.384314	0.379739	0.379739	0.379739	0.374510	0.374510	0.3
9	0.133333	0.133333	0.133333	0.133333	0.133333	0.133333	0.136601	0.136601	0.136601	0.149020	...	0.211111	0.216993	0.216993	0.216993	0.227451	0.227451	0.2
10	0.996078	0.996078	0.996078	0.998039	0.998039	0.998039	1.000000	1.000000	1.000000	0.996078	...	0.474510	0.479739	0.479739	0.479739	0.488235	0.488235	0.4

Figure 14: Template information in table format

Description:

The Iris template image which is converted as array is being stored in the table format.

PHASE 4: MODEL BUILDING

In this phase, the generated template data is used to train the set of models namely CNN and logistic regression. The template dataset is split in the ratio of 80-20 for train and test data. Both models are trained using train data and test data.

Figure 15 shows the model accuracy and loss while training the CNN model without max pooling.

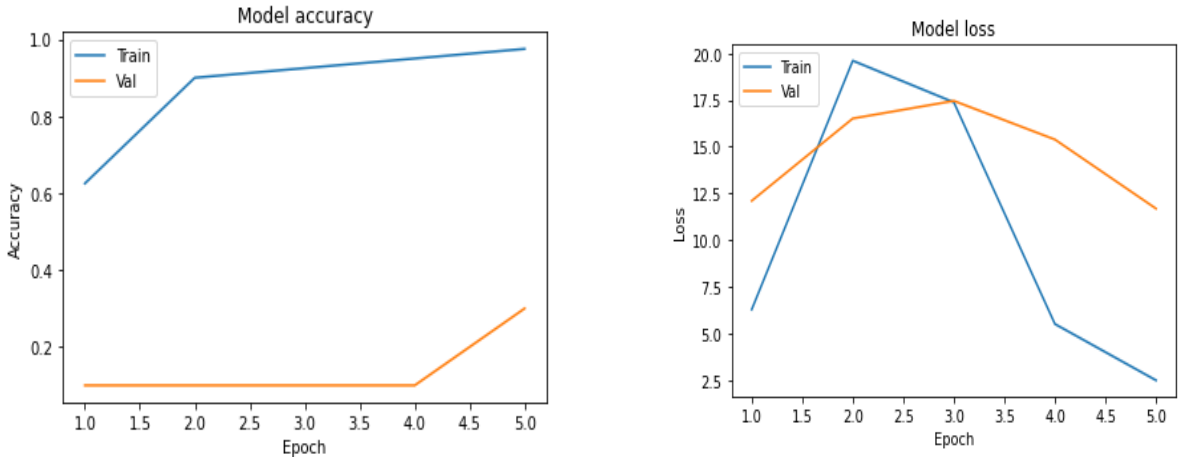


Figure 15: CNN model accuracy and loss using train data without max pooling

From figure 15, it is observed that the CNN model achieves maximum of 97.5% accuracy with minimal loss to detect and classify the template attack and genuine attack.

Figure 16 shows the model accuracy and loss while training the CNN model with max pooling.

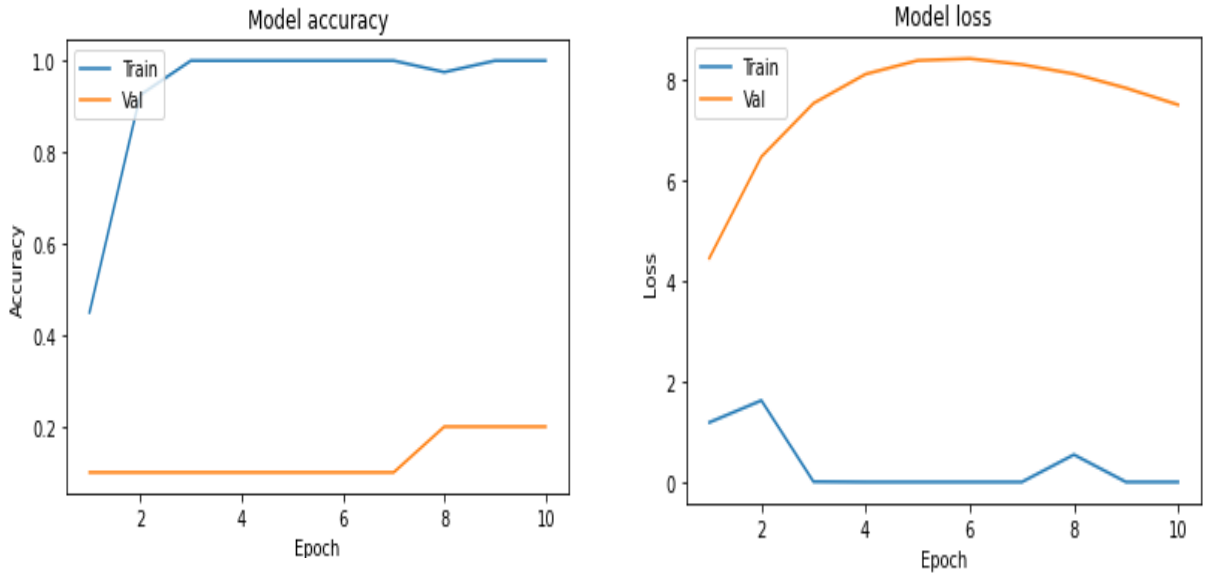


Figure 16: CNN model accuracy and loss using train data with max pooling

From figure 16, it is observed that the CNN model achieves maximum of 100% accuracy with minimal loss to detect and classify the template attack and genuine attack with max pooling. Now, logistic regression model is trained using train data and test using test data to detect and classify the attack template with genuine template. To evaluate the performance of LR model, the confusion matrix is calculated.

Figure 17 shows the confusion matrix and other evaluation metrics of the LR model.

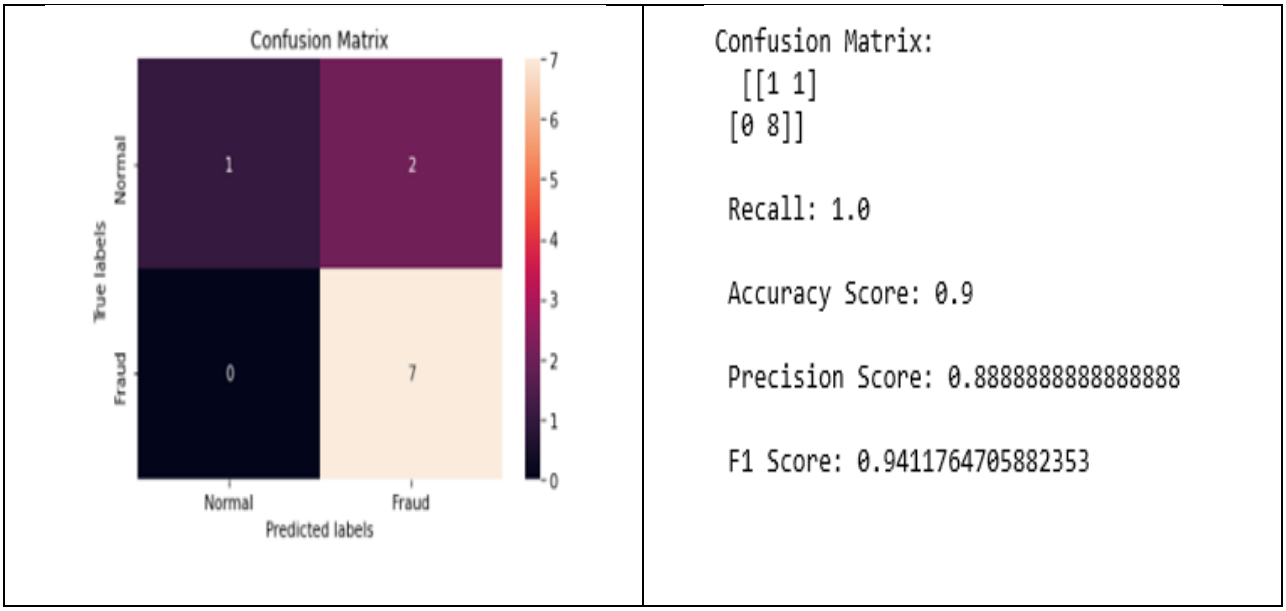
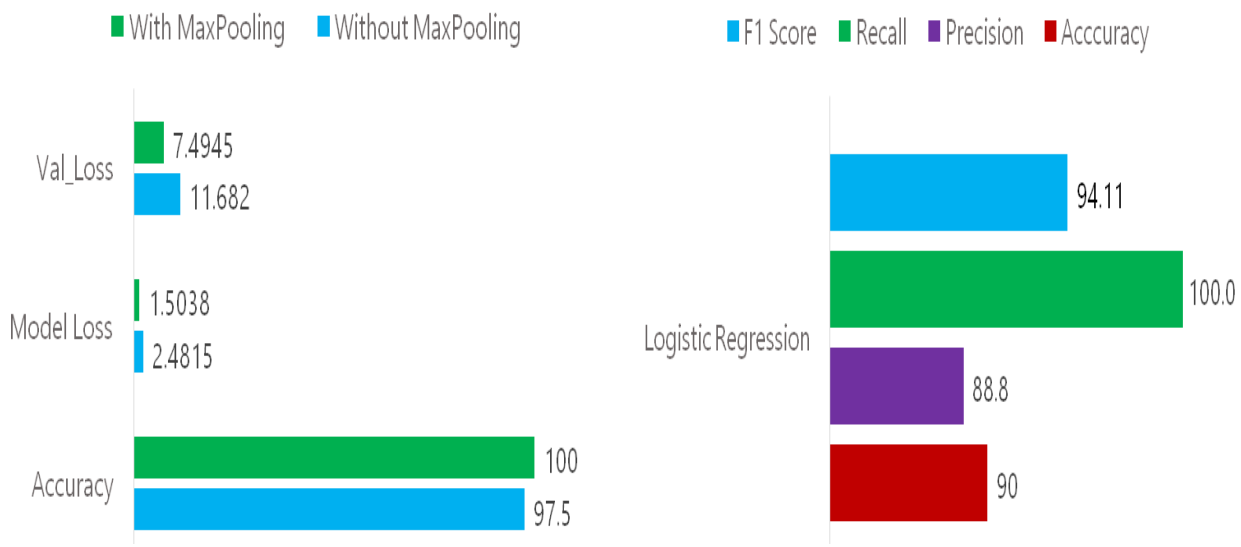


Figure 17: Evaluation of LR Model

It is observed that the LR model is able to detect the entire attack template but it fails to detect one genuine template and achieves the precision score of 89%. It achieves 90% accuracy and 100% recall with 94.11% F1 score.

PHASE 5: PERFORMANCE EVALUATION

Figure 18 shows the performance evaluation of CNN and LR to detect and classify the attack template and genuine template.



(a) CNN (b) Logistic Regression

Figure 18: Performance Evaluation of CNN and LR.

It is observed that the CNN model with max pooling achieves 100% accuracy with minimal loss than the CNN model without max pooling. The LR model achieves higher recall with huge F1 score and accuracy.

CONCLUSION AND FUTURE SCOPE

5. CONCLUSION AND FUTURE SCOPE

The attack template and genuine template is detected and classified using CASIA iris dataset. This image dataset successfully pre-processed using pre-processing techniques such as eye detection using cascade classifier, iris detection using Hough circle, morphological operation, edge contour detection and iris segmentation using Hough transform. This segmented image for CASIA iris image is considered as genuine template. The attack is successfully performed in genuine template and stored it as attacked template. Models namely Convolutional Neural Network and Logistic Regression are successfully trained using train and test data of both templates. it is observed that both the model performs better and achieves higher accuracy to detect and classify the genuine template and attack template. it is concluded that Machine Learning technique using Logistic Regression and Deep Learning technique using Convolutional Neural Network are explored successfully and obtains higher accuracy in detecting and classifying the genuine template and attacked template using CASIA iris image.

In future, other algorithms in Deep Learning such as, Self Organizing Maps, Generative Adversarial Networks, etc.., can be explored for detecting and classifying the genuine template and attacked template using CASIA iris dataset.

REFERENCES

6. REFERENCES

[1] B. E. D. A. D. Fatima, A. D. J. O. U. D. J. Réda and B. O. U. S. A. H. B. A. Nassima. "Studies of the Robustness of a Transformation-Based Multi-Biometric Template Schemes Protection," Int. J. Com. Dig. Sys. 2022, 11(1).

[2] M. Gupta and P. Sehgal. "HsIrisNet: Histogram based Iris recognition to allay replay and template attack using deep learning perspective," Pattern Recognition and Image Analysis. 2020, 30(4), pp. 786-794.

[3] J. McGrath, K. W. Bowyer and A. Czajka, "Open source presentation attack detection baseline for iris recognition," arXiv preprint arXiv:1809.10172. 2018.

[4] V. K. Sinha, A. K. Gupta and R. Khanna. "Detection of Fake Iris by using Frame Difference and Reflection Ratio."

[5] R. Sujitha and N. Lalithamani. "Counter Measures for Indirect Attack for Iris based Biometric Authentication," Indian Journal of Science and Technology. 2016, 9(19), pp. 1-7.

[6] M. Gomez-Barrero, J. Galbally, P. Tome and J. Fierrez. "On the vulnerability of iris-based systems to a software attack based on a genetic algorithm," Iberoamerican Congress on Pattern Recognition Springer, Berlin, Heidelberg. 2012, September, . pp. 114-121.

[7] C. Rathgeb and A. Uhl. "Statistical attack against iris-biometric fuzzy commitment schemes," CVPR 2011 WORKSHOPS. IEEE. 2011, June. pp. 23-30.

[8] J. Daugman. "How iris recognition works The essential guide to image processing" Academic Press. 2009, pp. 715-739.

[9] S. Sanderson and J. Erbetta. "Authentication for secure environments based on iris scanning technology," IEE Colloquium on Visual Biometrics. 2000.

[10] E. Wolff. “Anatomy of the Eye and Orbit,” 7th edition. H. K. Lewis & Co. LTD. 1976.

[11] R. Wildes. “Iris recognition: an emerging biometric technology,” Proceedings of the IEEE. 1997, 85(9).

[12] J. Daugman. “Biometric personal identification system based on iris analysis,” United States Patent, 1994, Patent Number: 5,291,56.

[13] J. Daugman. “High confidence visual recognition of persons by a test of statistical independence,” IEEE Transactions on Pattern Analysis and Machine Intelligence. 1993, 15(11).

[14] R. Wildes, J. Asmuth, G. Green, S. Hsu, R. Kolczynski, J. Matey and S. McBride. “A system for automated iris recognition,” Proceedings IEEE Workshop on Applications of Computer Vision. Sarasota, FL. 1994, pp. 121-128.

[15].https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

ANNEXTURE

ANNEXURE

SAMPLE CODING

```
import cv2
import numpy as np
import glob
import pickle


eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
eye_num_2=0
def transform_image(img,threshold):
    retval, threshold = cv2.threshold(img, threshold, 255, cv2.THRESH_BINARY)
    opening = cv2.morphologyEx(threshold, cv2.MORPH_OPEN, kernel)
    closing = cv2.morphologyEx(threshold, cv2.MORPH_CLOSE, kernel)
    open_close = cv2.bitwise_or(opening, closing, mask = None)
    return open_close,opening,closing


imgs = []
label=0
final_output = []
lables = []
eye_detected = []
```

```
iris_eye_detected=[]

for filepath in glob.iglob('CASIA1/*'):
    num_in_folder=0

    img = cv2.imread(filepath)
    img=cv2.resize(img,(200,150))
    img =        cv2.cvtColor(img,    cv2.COLOR_BGR2GRAY)
    imgs.append([img,num_in_folder,label,img])
    print(filepath)
    num_in_folder = num_in_folder+1


    label=label+1
    #print(filefilepath)

print("total images number ",len(imgs))

eyes_num=0
for i,j,L,c in imgs:

    # cv2.imshow('dd',i)
```

```
    i=cv2.resize(i,(400,300))
    eyes = eye_cascade.detectMultiScale(i, 1.01, 0)

    if len(eyes)>1:
        eye_detected.append(imgs[eyes_num])
        print(eyes_num)
        eyes_num = eyes_num+1
        maxium_area = -3
        for (ex,ey,ew,eh) in eyes:
            area = ew*eh
            if area>maxium_area:
                maxium_area = area
                maxium_width=ew
                point_x=ex
                point_y=ey
                maxium_height = eh

cv2.rectangle(i,(point_x,point_y),(point_x+maxium_width,+maxium_height),(255,0,0),2)
        cv2.imwrite('Template/2/eye_detection/'+str(L)+'.'+str(j)+'.jpg',i)

print("total_eyes_found = ",eyes_num)
```

```
iris_num=0
for i,j,L,c in eye_detected:

    circles = cv2.HoughCircles(i, cv2.HOUGH_GRADIENT, 10, 100)
    if circles is not None :
        circles = np.round(circles[0, :]).astype("int")
        #print(len(circles))
        #print(y)

        maxiumum_average=10000000000000
        #print(len(circles))
        print(i.shape[0])
        print(i.shape[1])
        print(min(i.shape))

        key=True
        for (x, y, r) in circles:

            if x+r<=max(i.shape) and y+r<=max(i.shape)and x-r>0 and y-r>0 and r>20:
                key=False
```

```
        new_roi = i[y-r:y+r, x-r:x+r]
        average = np.average(new_roi)

        if average < maxiumum_average:
            maxiumum_r = r
            point_x=x
            point_y=y
            maxiumum_average=average

        #cv2.circle(i, (x, y), r, (0, 0, 0), 4)

if key:
    #print("key opened")

    for (x, y, r) in circles:
            maxiumu_raduis=-4
            if r > maxiumu_raduis:
                maxiumum_r = r
                point_x=x
                point_y=y
                #maxiumum_average=average
```

```
        cv2.circle(i, (point_x, point_y), maxiumum_r, (255, 255, 0), 4)
        cv2.imwrite('Template/2/iris_detection/'+str(L)+'.'+str(j)+'.jpg',i)

iris_eye_detected.append(eye_detected[iris_num])
        print(iris_num)
        iris_num = iris_num+1

print("total_iris_found = ",iris_num)
print("total images number ",len(imgs))

imgs= iris_eye_detected

kernel = np.ones((5,5),np.uint8)
import random

random.shuffle(imgs)

test=[]
for i,j,L,c in imgs:

    gold,siver,diamond = transform_image(i,0)
    golden_refrence = sum(sum(gold))
```

```
    found = True

    for k in range(10,10000,10):

        working_img,opening,closing = transform_image(i,k)
        suming = sum(sum(working_img))
        diffrence = suming-golden_refrence

        if diffrence>800:
            found = False
            print("the image threshold = " ,k)
            print("the image name " +str(j))
            print(" " )

            contours,_ = cv2.findContours(working_img, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

            cv2.imwrite('Template/2/threshold/'+str(L)+'.'+str(j)+'.jpg',working_img)
            cv2.imwrite('Template/2/opening/'+str(L)+'.'+str(j)+'.jpg',opening)
            cv2.imwrite('Template/2/closing/'+str(L)+'.'+str(j)+'.jpg',closing)

            for z in contours:
```

```
            x,y,w,h = cv2.boundingRect(z)
            if x+w<150 and y+h<200 and x-w//4>0:


                cv2.rectangle(working_img,(x,y),(x+w,y+h),(0,255,0),-2)
                cv2.imwrite('Template/2/contour/'+str(L)+'.'+str(j)+'.jpg',working_img)


                 contours_2,_ = cv2.findContours(working_img, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
            maxium_area=0
            maxium_area = 0
            maxium_width=0
            point_x=0
            point_y=0
            maxium_height = 0
            for z in contours_2:
               #print(len(i))
               x,y,w,h = cv2.boundingRect(z)
               new_area=h*w
               if x+w<150 and y+h<200 and new_area>maxium_area and x-w//4>0:
                  maxium_area = new_area
                  maxium_width=w
                  point_x=x
```

```
            point_y=y
            maxium_height = h

        center_x = point_x+maxium_width//2
        center_y = point_y+maxium_height//2
        radius = 40

        if center_y-radius>0 and center_x-radius >0  and center_y+radius < 200 and
center_x+radius < 150:
             new_roi = c[center_y-radius:center_y+radius, center_x-
radius:center_x+radius]
            new_roi=cv2.resize(new_roi,(200,150))

            cv2.imwrite('Template/2/final_casia/'+str(L)+'.'+str(j)+'.jpg',new_roi)

else:
            center_y=c.shape[0]//2
            center_x=c.shape[1]//2
            new_roi = c[center_y-radius:center_y+radius, center_x-radius:center_x+radius]
            new_roi =cv2.resize(new_roi,(200,150))
            #new_roi = cv2.cvtColor(new_roi,cv2.COLOR_GRAY2BGR)

            cv2.imwrite('Template/2/final_casia/'+str(L)+'.'+str(j)+'.jpg',new_roi)
```

```
            cv2.imwrite('Template/2/edging_5/'+str(L)+'_'+str(j)+'.jpg',i)
            test.append(i)
            final_output.append(new_roi)
            lables.append(L)

print("the lenght of final output = ",len(final_output))
print("the of lables = ",len(lables))

final_output
img_size=(100,100)
y=tensorflow.keras.utils.to_categorical(lables, num_classes=1000, dtype='float32')
final_output = np.array(final_output,dtype="float16")/255

img_size, final_output[0]

test = final_output[0]
test = test.reshape((1, img_size[0], img_size[1],3))
test_shape = bottleneck_model.predict(test).shape
print(test_shape)
shape = (final_output.shape[0],test_shape[1],test_shape[2],test_shape[3])
print(shape)
```

```
print(shape[1:])


bottelneck_features = []
for i in final_output:
        i = i.reshape((1, img_size[0], img_size[1],3))
    bottelneck_features.append(bottleneck_model.predict(i))
    print(len(bottelneck_features))


bottelneck_features=np.array(bottelneck_features)
print(bottelneck_features.shape)


bottelneck_features
bottelneck_features =  bottelneck_features.reshape(shape)
print(bottelneck_features.shape)


import pandas as pd
import os
from skimage.transform import resize
from skimage.io import imread
import numpy as np
```

```
import matplotlib.pyplot as plt


import os
Categories=['attacked template', 'original template']


flat_data_arr=[] #input array
target_arr=[] #output array


datadir= 'C:/Users/Admin/anaconda3/Iris (2)/Iris/category/'
for i in Categories:
    print(f'loading... category : {i}')
    path=os.path.join(datadir,i)
    for img in os.listdir(path):
        img_array=imread(os.path.join(path,img))
        img_resized=resize(img_array,(150,150,3))
        flat_data_arr.append(img_resized.flatten())
        target_arr.append(Categories.index(i))
    print(f'loaded category:{i} successfully')
```

```
flat_data=np.array(flat_data_arr)

target=np.array(target_arr)

df=pd.DataFrame(flat_data) #dataframe

df['Target']=target

x=df.iloc[:,:-1] #input data

y=df.iloc[:,-1] #output data


from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)

print('Splitted Successfully')

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(x_train)

X_test = scaler.transform(x_test)


y_train = y_train.to_numpy()

y_test = y_test.to_numpy()
```

```
X_train.shape


X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)

X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

X_train.shape, X_test.shape


#Build CNN

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Flatten, Dense, Dropout, BatchNormalization

from tensorflow.keras.layers import Conv1D, MaxPool1D

from tensorflow.keras.optimizers import Adam

print(tf.__version__)


epochs = 5

model = Sequential()

model.add(Conv1D(28, 2, activation='relu', input_shape = X_train[0].shape))
```

```
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Conv1D(60, 2, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(60, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.summary()

model.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy', metrics=['accuracy'])
def plot_learningCurve(history, epoch):
  # Plot training & validation accuracy values
    epoch_range = range(1, epoch+1)
    plt.plot(epoch_range, history.history['accuracy'])
    plt.plot(epoch_range, history.history['val_accuracy'])
```

```
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Val'], loc='upper left')
    plt.show()

  # Plot training & validation loss values
    plt.plot(epoch_range, history.history['loss'])
    plt.plot(epoch_range, history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Val'], loc='upper left')
    plt.show()

plot_learningCurve(history, epochs)

#Adding MaxPool
```

```
epochs = 10

model = Sequential()

model.add(Conv1D(28, 2, activation='relu', input_shape = X_train[0].shape))

model.add(BatchNormalization())

model.add(MaxPool1D(2))

model.add(Dropout(0.2))


model.add(Conv1D(60, 2, activation='relu'))

model.add(BatchNormalization())

model.add(MaxPool1D(2))

model.add(Dropout(0.5))


model.add(Flatten())

model.add(Dense(60, activation='relu'))

model.add(Dropout(0.5))


model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy',
metrics=['accuracy'])
```

```
history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test, y_test), verbose=1)

plot_learningCurve(history, epochs)


import pandas as pd

import os

from skimage.transform import resize

from skimage.io import imread

import numpy as np

import matplotlib.pyplot as plt


import os

Categories=['attacked template', 'original template']


flat_data_arr=[] #input array

target_arr=[] #output array


datadir= 'C:/Users/Admin/anaconda3/Iris (2)/Iris/category/'
```

```
for i in Categories:
    print(f'loading... category : {i}')
    path=os.path.join(datadir,i)
    for img in os.listdir(path):
        img_array=imread(os.path.join(path,img))
        img_resized=resize(img_array,(150,150,3))
        flat_data_arr.append(img_resized.flatten())
        target_arr.append(Categories.index(i))
    print(f'loaded category:{i} successfully')
flat_data=np.array(flat_data_arr)
target=np.array(target_arr)
df=pd.DataFrame(flat_data) #dataframe
df['Target']=target
x=df.iloc[:,:-1] #input data
y=df.iloc[:,-1] #output data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
print('Splitted Successfully')
```

```
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import GridSearchCV

logmodel= LogisticRegression(random_state=0)


bestlogcls_us = logmodel.fit(x_train, y_train)


y_pred=logmodel.predict(x_test)

y_pred, y_test


metrics.accuracy_score(y_test, y_pred)

from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score, recall_score, classification_report, roc_curve, auc

def model_performance(y_test, y_pred):

    print(f'Confusion Matrix:\n ',confusion_matrix(y_test, y_pred))

    print(f'\n Recall:', recall_score(y_test, y_pred))

    print(f'\n Accuracy Score:', accuracy_score(y_test, y_pred))

    print(f'\n Precision Score:', precision_score(y_test, y_pred))

    print(f'\n F1 Score:' ,f1_score(y_test, y_pred))

    return
```

```
predict_us = bestlogcls_us.predict(x_test)

model_performance(y_test,predict_us)


from sklearn import metrics

import matplotlib.ticker as ticker

import seaborn as sns


conf_matrix = metrics.confusion_matrix(y_test,y_pred)


ax=plt.subplot()

sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')#annot=True to annotate cells, fmt='g' numbers not scientific form

ax.set_xlabel('Predicted labels'); ax.set_ylabel('True labels')

ax.set_title('Confusion Matrix');

ax.xaxis.set_ticklabels(['Normal', 'Fraud']); ax.yaxis.set_ticklabels(['Normal', 'Fraud']);

ax.set(yticks=[0, 2],

      xticks=[0.5, 1.5])

ax.yaxis.set_major_locator(ticker.IndexLocator(base=1, offset=0.5))
```