DETECTING FINGERPRINT SPOOFING USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

DIVYA S (910619104017)

GOPIKA R (910619104022)

MANISHA G (910619104044)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K.L.N. COLLEGE OF ENGINEERING, POTTAPALAYAM

(An Autonomous Institution, Affiliated to Anna University, Chennai)

ANNAUNIVERSITY: CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“Detecting Fingerprint Spoofing Using Machine Learning”** is the bonafide work of **“DIVYA S (910619104017)”**, **“GOPIKA R (910619104022)”**, **“MANISHA (910619104044)”**, who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.S.MIRUNA JOE AMALI	Mr.D.PRAVIN KUMAR
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	ASSOSIATE PROFESSOR
Computer Science and Engineering K.L.N. College of Engineering, Pottapalayam, Sivagangai-630 612.	Computer Science and Engineering K.L.N. College of Engineering, Pottapalayam, Sivagangai-630 612.

Submitted for the project viva-voce conducted on _______________.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be but incomplete without the mention of the people made it possible, whose constant guidance and encouragement crowned our efforts with success.

We extend our gratitude to the Founder, late**, Thiru.K.L.N.KRISHNAN,** K.L.N College of Engineering and Management Members for making us march towards the glory of success. We express our sincere thanks to our respected Principal **Dr.A.V. RAM PRASAD, M.E, Ph.D., MISTE.FIE,** for all the facilities offered.

We would like to express our deep gratitude and heartfelt thanks to **Dr.S.MIRUNA JOE AMALI, M.E, Ph.D.,** Head of the Department of Computer Science and Engineering, K.L.N College of Engineering, who motivated and encouraged us to do this out rival project for this academic year.

Our profound, delightful and sincere thanks to our Project Guide **Mr.D.PRAVIN KUMAR,M.E., MBA.,** and our project coordinator **Mrs.V.KARPAGAM, M.E.,** and Head of the department,
Dr.S. MIRUNA JOE AMALI, M.E., Ph.D., whose support was inevitable during the entire period of our work. We thank our teaching staffs for sharing their knowledge and view to enhance our project.

We also thank our non-teaching staff for extending their technical support to us. I thank my parents for giving me such a wonderful life and my friends for their friendly encouragement throughout the project. Finally, we thank the Almighty for giving the full health to finish the Project successfully.

ABSTRACT

As a result of the growing use of commercial fingerprint authentication systems in mobile devices, detection of fingerprint spoofing has become increasingly important and widely used. The features of fingerprint liveness-detection method, based on convolutional neural network (CNN) was extracted from fingerprint patches. Firstly, fingerprints are segmented, and then data augmentation is performed to increase the size of training data. Secondly, on the augmented fingerprint, locations of patches are determined through normal distributions of segmented areas of the fingerprint image. It is used to prevent from unauthorized access. It helps in preventing intruder's attack to deceive the device. It provides more accuracy in terms of spoof detection, which is an open-set method. A self-learning, secure and independent open-set solution is essential to be explored to characterize the liveness of fingerprint presentation. Fingerprint spoof presentation classified as live (a Type-I error) is a major problem in a high-security establishment. Type-I error are manifestation of small number of spoof sample. The system proposes to use only live sample to overcome above challenge. The system put forward an adaptive 'fingerprint presentation attack detection' (FPAD) scheme using interpretation of live sample. It requires initial high-quality live fingerprint sample of the concerned person. It uses six different image quality metrics as a transient attribute from each live sample and records it as 'Transient Liveness Factor' (TLF). Therefore the study also proposes to apply fusion rule to validate scheme with three outlier detection algorithms, one-class Convolutional Neural Network (CNN), **Isolation** Forest and Local Outlier actor. Proposed study got phenomenal accuracy of 99% in terms of spoof detection, which is an open-set method using CNN. Further, the study proposes and discusses open issues on person specific spoof detection on cloud-based solutions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSION
CNN	Convolutional Neural Network
FPAD	Fingerprint Presentation Attack Detection
TLF	Transient Liveness Factor
PHP	Hypertext Preprocessor
SQL	Structured Query Language.
GB	Giga Byte
LIVDET	Liveness Detection

CHAPTER 1

1. INTRODUCTION

1.1. PURPOSE

Nowadays, biometric recognition systems have used in a variety of identification sectors, due to their convenience and robustness compared with conventional techniques such as a password. Biometrics recognition systems rely on the physiological and behavioral attributes of individuals. The fingerprint is one of the most frequently used authentication systems since they guarantee high identification accuracy, cost-effective, and can be applied to huge datasets of images. Those characteristics make fingerprint recognition systems deployed in many applications, such as attendance, smart phone identification, forensics, health-care systems, banks, etc. However, those systems are not aloof from malicious attacks.

There are two types of attacks that biometric are vulnerable from direct, and indirect attacks. Direct attack is the most common since there is no knowledge is required to conduct the attack. For the fingerprint recognition system, it can be performed in the sensor device with simple and handy tools like silicon, play-doh, wood glue, etc...In contrast, indirect attack imposes deep information about the system's module. With the increased amount of attack tools, researchers have attracted to develop a system that can assess and provide a solution for liveness detection of fingerprint systems. It shows the researches documents proposed in fingerprint biometrics which rapidly growing and attract the researchers in the last years.

1.1.1. PROJECT SCOPE

To verify the fingerprints using machine learning algorithm like Convolutional Neural Network. It is also used,

- to check the spoof finger print.
- to provide more security, accuracy and effective result.
- The proposed system produce liveness fingerprint detection schemes.
- It reviews that many studies proposed in liveness fingerprint detection systems utilize CNN.

1.2. OVERVIEW OF CNN ALGORITHM

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.

CNNs are trained using a large dataset of labeled images, where the network learns to recognize patterns and features that are associated with specific objects or classes. Once trained, a CNN can be used to classify new images, or extract features for use in other applications such as object detection or image segmentation.

CNNs have achieved state-of-the-art performance on a wide range of image recognition tasks, including object classification, object detection, and image segmentation. They are widely used in computer vision, image processing, and other related fields, and have been applied to a wide range of applications, including self-driving cars, medical imaging, and security systems.

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision.

CNN can run directly on a underdone image and do not need any preprocessing. A convolutional neural network is a feed forward neural network, seldom with up to 20. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer.

CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces.

The agenda for this sphere is to activate machines to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order. It is the sequential design that give permission to CNN to learn hierarchical attributes.

CHAPTER 2

2. SYSTEM ANALYSIS

2.1. LITERATURE SURVEY

[1] R.P.Krish et al.,"Improvingautomatedlatent fingerprint identification using extended minutia types", InformationFusion, Vol.50, pp.9-19, 2020.

The paper "Improving automated latent fingerprint identification using extended minutia types" by R.P.Krish et al. was published in the journal Information Fusion in 2020. The paper addresses the problem of automated latent fingerprint identification, which involves matching an unknown or latent fingerprint against a database of known fingerprints.

The authors propose the use of extended minutia types, which are additional features that can be extracted from fingerprints to improve identification accuracy. These extended minutia types include delta points, core points, and ridge counts, which are commonly used in fingerprint analysis.

The authors conducted experiments to evaluate the performance of their proposed method using a publicly available dataset. The results showed that the use of extended minutia types improved the identification accuracy compared to using traditional minutia features alone.

The paper also discusses the limitations of the proposed method, such as the sensitivity of the method to the quality of the fingerprint image, and the need for additional research to optimize the selection of the extended minutia types.

Overall, the paper provides a promising approach for improving automated latent fingerprint identification, which has important applications in law enforcement and forensic science.

A linear binary pattern had used in the feature extraction phase, by resizing the fingerprint image to the size 60*60. The results demonstrated that the neural network over performed the nearest neighbor classifier in terms of accuracy.

[2] S. Khade, S. D. Thepade, and A. Ambedkar, "Fingerprint Liveness Detection Using Directional Ridge Frequency with Machine Learning Classifiers", in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).pp.1-5, 2020.

The paper "Fingerprint Liveness Detection Using Directional Ridge Frequency with Machine Learning Classifiers" by S. Khade, S. D. Thepade, and A. Ambedkar was presented at the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

The paper addresses the problem of fingerprint liveness detection, which is the ability to distinguish between live and fake or spoofed fingerprints. The authors propose the use of directional ridge frequency (DRF), a feature extraction method that analyzes the frequency of the ridges in a fingerprint image in different directions, to detect liveness.

The authors conducted experiments to evaluate the performance of their proposed method using a publicly available dataset. They also compared their method with other existing methods for liveness detection. The results showed that the use of DRF with machine learning classifiers improved the accuracy of liveness detection compared to other methods.

The paper also discusses the limitations of the proposed method, such as the need for a high-quality fingerprint image and the sensitivity of the method to changes in the imaging conditions.

Overall, the paper provides a promising approach for fingerprint liveness detection, which has important applications in biometric authentication and security.

They attempted to enhance the latent fingerprint identification. The experiments of their method had applied using a realistic forensic fingerprint dataset and tested with three minutiae-based matches. The result obtained an improvement in identification accuracy.

[3] Q. Huang, S. Chang, C. Liu, B. Niu, M. Tang, and Z. Zhou, "An evaluation o f fake fingerprint datasets utilizing SVM classification", Patt. Recogn. Lett., pp.1-7, 2019.

The article "An evaluation of fake fingerprint datasets utilizing SVM classification" by Huang, Chang, Liu, Niu, Tang, and Zhou is a research paper that discusses the evaluation of fake fingerprint datasets using Support Vector Machine (SVM) classification.

The study aims to evaluate the performance of SVM classification on various fake fingerprint datasets, including the LivDet 2013 and 2015 datasets, and to investigate the impact of different factors, such as the type of fake fingerprint and the number of training samples, on the performance of the classifier.

The researchers used a range of evaluation metrics, including the Equal Error Rate (EER), the False Acceptance Rate (FAR), and the False Rejection Rate (FRR), to assess the performance of the SVM classifier on the different datasets. They also compared the results with other state-of-the-art techniques in the field of fake fingerprint detection.

Overall, the study found that SVM classification was an effective technique for detecting fake fingerprints, particularly when using a large number of training samples. The researchers also found that the type of fake fingerprint had a significant impact on the performance of the classifier, with some types of fakes being more challenging to detect than others.

The study concludes that SVM classification is a promising approach for fake fingerprint detection, and that further research is needed to develop more accurate and robust detection techniques. SVM method had proposed for feature extraction based on directional ridge frequency. All the directions had considered: horizontal, vertical, forward diagonal and backward diagonal to get better accuracy.

2.2. EXISTING SYSTEM

Identification of a live fingerprint image is a major challenge nowadays. Earlier researchers proposed approaches on spoof detection using close-set methods. These methods bound them to fail under a certain condition. One of limitation is presence of Type-I error, spoof classified as live, which is not good for critical system. In recent history, FPAD implementation came out in a variety of the form. Reported studies suggest a threefold increase in the error rates of fingerprint spoof detectors when spoofs using new materials come during the testing or operational stage. This means the generalization capability of existing fingerprint spoof detectors is limited across materials. Deep Learning is used to examine fingerprint Presentation Attack Detection (PAD) in depth. The examination clearly states that software based PAD techniques are more efficient as compare to hardware.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

- In preceding implementation, they have only bio-metric device to check authorized person or not. It will be verified with database.
- But intruders may do the practice of duplicate finger print copy with respective material.
- It will be working and verify that person as authorized.
- It may lead several security problems.
- Deep Learning requires large volume of data and are not as trained as Machine Learning.
- It lacks live object detection for differentiating original and spoof fingerprints.

2.3. PROPOSED SYSTEM

With the steady advancement of attack modes, the attacks come in ‘realistic-looking’ and ‘End-less variety’ form known as fake fingerprint presentations. Unlike the existing schemes, the study uses person-specific live sample to extract liveness feature. The live-sample has inherent liveness feature, with multiple live samples collected during enrollment or over a period, undergoes for measuring liveness aspects of each finger, calling it as ‘Transient Live Feature’.

To build the fingerprint system to be more reliable, many live fingerprint samples will achieve superior. Here, the liveness characteristics of the different live samples of an individual are taken from the standard dataset, several sets of independent properties are measured. Finally these features are correlated with traditional machine learning approaches. It guarantees high identification accuracy, cost-effective, and can be applied to huge datasets of images. Using CNN algorithm, the differentiation between liveness image and spoofing image based on threshold can be identified.

CNNs are trained using a large dataset of labeled images, where the network learns to recognize patterns and features that are associated with specific objects or classes. Once trained, a CNN can be used to classify new images, or extract features for use in other applications such as object detection or image segmentation. LivDet dataset is used to collect fingerprint PAD methods using implementation of standard testing metrics. CNNs have achieved state-of-the-art performance on a wide range of image recognition tasks, including object classification, object detection, and image segmentation..

2.4. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.

2.4.1. ECONOMICAL FEASIBILITY

An organization makes good investment on the system. So, they should be worthful for the amount they spend in the system. Always the financial benefit and equals or less the cost of the system, but should not exceed the cost.

- The cost of investment is analyzed for the entire system
- The cost of Hardware and Software is also noted.
- Analyzing the way in which the cost can be reduced

Every organization wants to reduce there cost but at the same time quality of the Service should also be maintained. The system is developed according the estimation of the cost made by the concern. In this project, the proposed system will definitely reduce the cost and also the manual work is reduced and speed of work is also increased.

2.4.2. TECHNICAL FEASIBILITY

The Technical feasibility is the study of the software and how it is included in the study of our project. Regarding this there are some technical issues that should be noted they are as follows:

- Is the necessary technique available and how it is suggested and acquired?
- Does the proposed equipment have the technical capacity to hold the data required using the new system?

- Will the system provide adequate response that is made by the requester at a periodic time interval
- Can this system be expanded after this project development
- Is there a technique guarantees of accuracy, reliability in case of access

The technical issues are raised during the feasibility study of investigating our System. Thus, the technical consideration evaluates the hardware requirements, software etc. This system uses PHP as front end and MySQL as back end. They also provide sufficient memory to hold and process the data. As the company is going to install all the process in the system it is the cheap and efficient technique.

This system technique accepts the entire request made by the user and the response is done without failure and delay. It is a study about the resources available and how they are achieved as an acceptable system. It is an essential process for analysis and definition of conducting a parallel assessment of technical feasibility.

Though storage and retrieval of information is enormous, it can be easily handled by MySQL. As the MySQL can be run in any system and the operation does not differ from one to another. So, this is effective.

2.4.3. SOCIAL FEASIBILITY

Proposed project will be beneficial only when they are turned into an information system and to meet the organization operating requirements. The following issues are considered for the operation:

- Does this system provide sufficient support for the user and the management?
- What is the method that should be used in this project?

- Have the users been involved in the planning and development of the projects?
- Will the proposed system cause any harm, bad result, loss of control and accessibility of the system will lost?

Issues that may be a minor problem will sometimes cause major problem in the operation. It is the measure of how people can able to work with the system. Finding out the minor issues that may be the initial problem of the system. It should be a user-friendly environment. All these aspect should be kept in mind and steps should be taken for developing the project carefully.

CHAPTER 3

SYSTEM SPECIFICATION

3.1. SOFTWARE SPECIFICATION

The below Software Specifications were used in both Server and Client machines when developing.

SERVER

Operating System	:	Windows 7
Technology Used	:	Python
Database	:	My-SQL
Database Connectivity	:	Native Connectivity
Web Server	:	Django
Browser	:	Chrome

CLIENT

Operating System	:	Windows 7
Browser	:	Chrome

3.2. HARDWARE SPECIFICATION

The below Hardware Specifications were used in both Server and Client machines when developing.

Processor	:	Intel(R) Core(TM) i7
Processor Speed	:	3.06 GHz
RAM	:	2 GB
Hard Disk Drive	:	250 GB
Floppy Disk Drive	:	Sony
CD-ROM Drive	:	Sony
Monitor	:	“17” inches
Keyboard	:	TVS Gold
Mouse	:	Logitech

CHAPTER 4

PROJECT DESCRIPTION

4.1. OVERVIEW OF THE PROJECT

We have given a scheme that can be extended as an architecture for liveness-server for extraction/updating transient liveness factor. We project, liveness identification is penultimate scope during design, development for testifying the FPAD system to a user. Efforts can used to model a system with generalization. We have discussed here a small step forward towards generalized and reproducible system.

This addresses a concern towards the requirement for the uniform system is need of today for future application. The approach proposed will act as a technology of future. A wider scope (usages) over smartphones and smart devices apart from conventional devices is possible then.

Only requirement is to design a liveness server according to end user machine(s). With this intelligence gained from TLF, the liveness server will become a new biometric technology backbone. Hackers will find it difficult to access any device. Because of this dynamism, the future machine may match the user access pattern together for more robust system.

4.2. PROJECT MODULES

1. LIVEDET DATASET
2. CROSS VALIDATION
3. CLASSIFICATION
4. TESTING

4.2.1 MODULES DESCRIPTION

LIVEDET DATASET

The process of a biometric system detecting a biometric spoof is known as Presentation Attack Detection (PAD). PAD systems utilize a combination of hardware and software technologies to determine whether a biometric system is genuine. A liveness detection is a subset of presentation attack detection. All the data during spoof detection through PAD method are stored in LIVDET server. From LIVDET server genuine and forged fingerprint images are collected for training and testing the model.

CROSS VALIDATION

During cross-validation the entire dataset will be divided into two sets, the training set and testing set. 80% of the dataset goes to the training set and 20% to the test set. The model will be trained on the training set and tested on the test set.

CLASSIFICATION

Using softmax classifier the fingerprints were classified as genuine or forged. Using collected finger print images from LiveDet, apply the CNN algorithm to predict the result and identify the finger print whether it is live or spoofing. A combination of datasets had used in the training model which help to improve the accuracy.

TESTING

CNN converts the fingerprint images into fingerprint patches. By analyzing the patches it decides whether it is spoofed or live fingerprint through more amount of training in the training process. As CNN converts fingerprint images into patches, it can predict the accurate result.

4.3. OVERVIEW OF TOOLS AND LANGUAGE

4.3.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is opensource software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

4.3.2 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

4.3.3 MYSQL

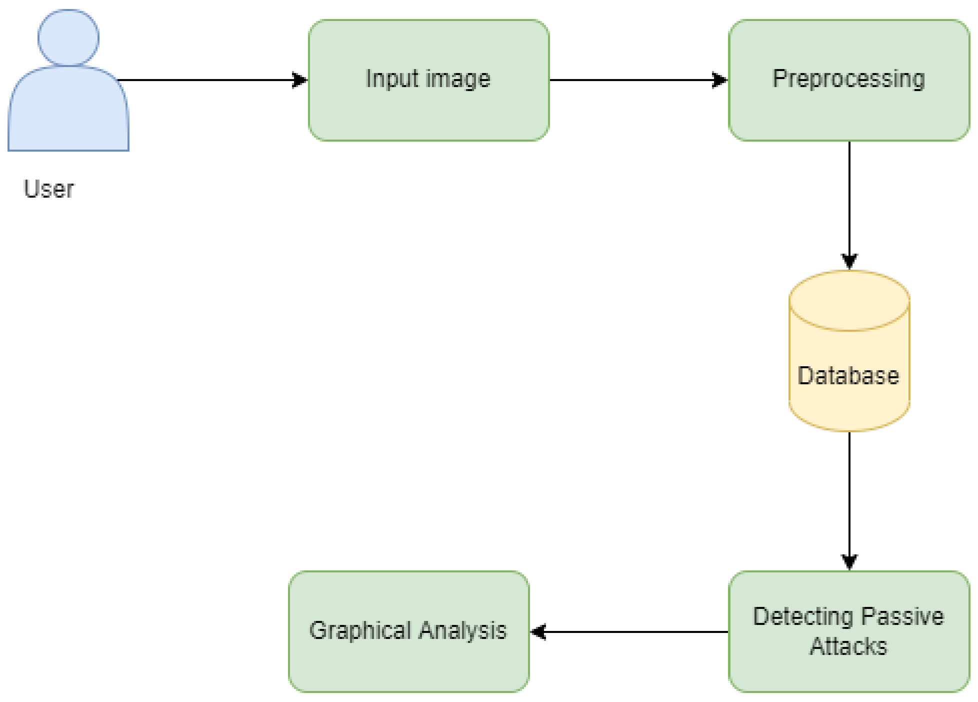
MySQL (http://www.mysql.com) is a robust SQL database server developed and maintained by T.c.XDataKonsultAB of Stockholm, Sweden. Publically available since 1995, MySQL has risen to become one of the most popular database servers in the world, this popularity due in part to the server’s speed, robustness, and flexible licensing policy. (See note for more information regarding MySQL’s licensing strategy.)

Given the merits of MySQL’s characteristics, coupled with a vast and extremely easy-to-use set of predefined interfacing functions, MySQL has arguably become PHP’s most-popular database counterpart.

CHAPTER 5

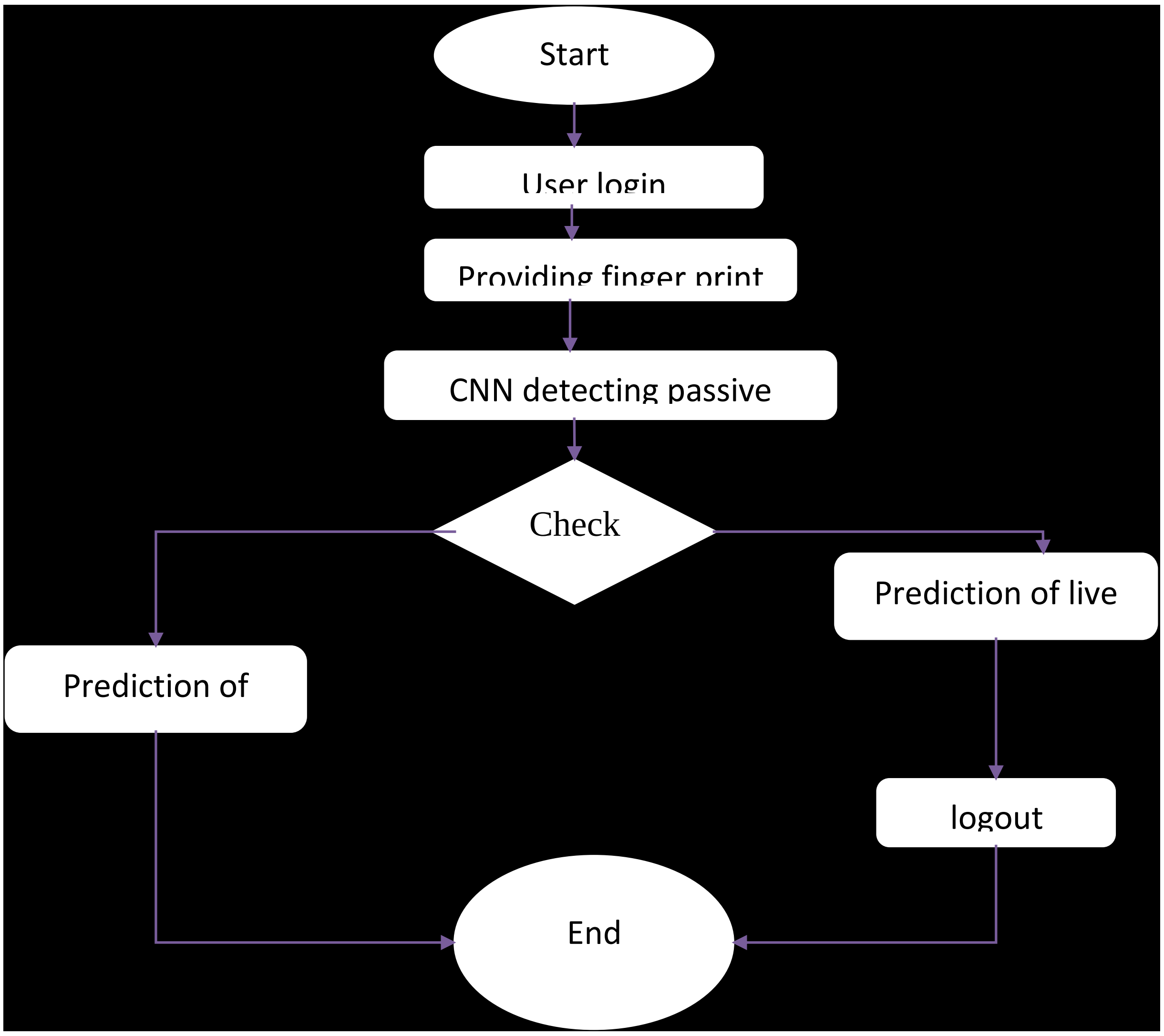
SYSTEM DESIGN

5.1 ARCHITECTURAL DESIGN



Figno:5.1 Architectural Diagram

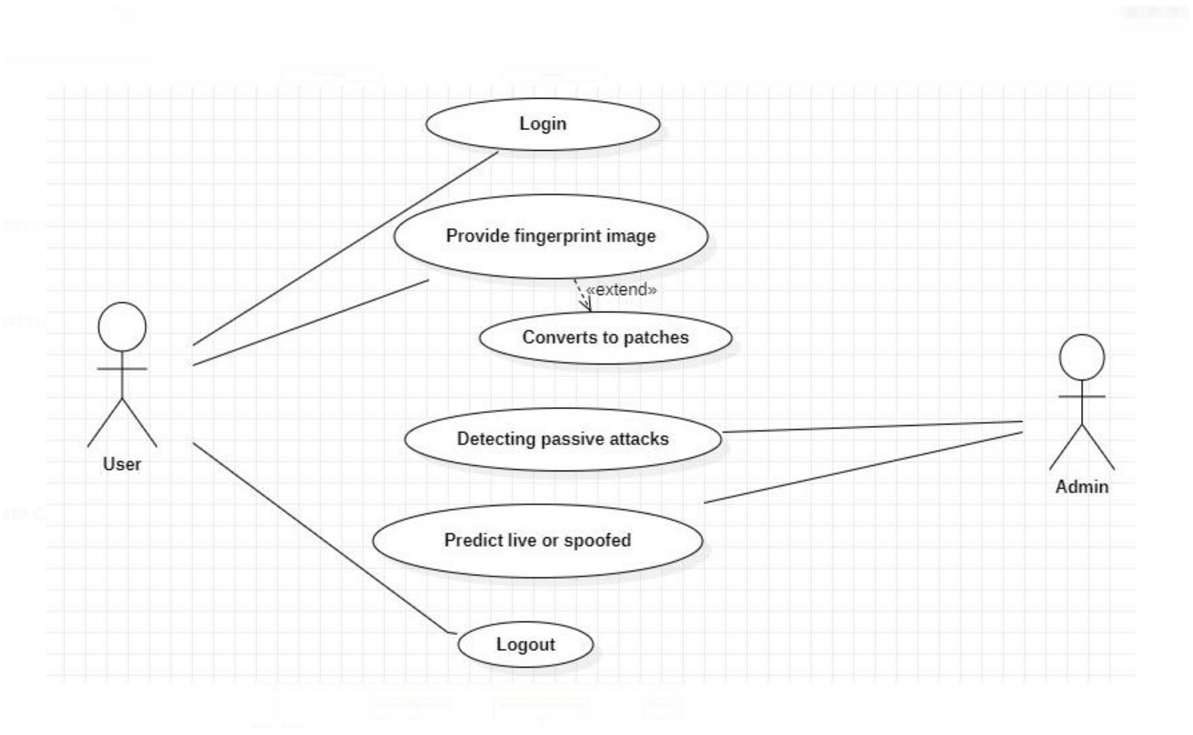
5.2 DATA FLOW DIAGRAM



Figno:5.2 Data Flow Diagram

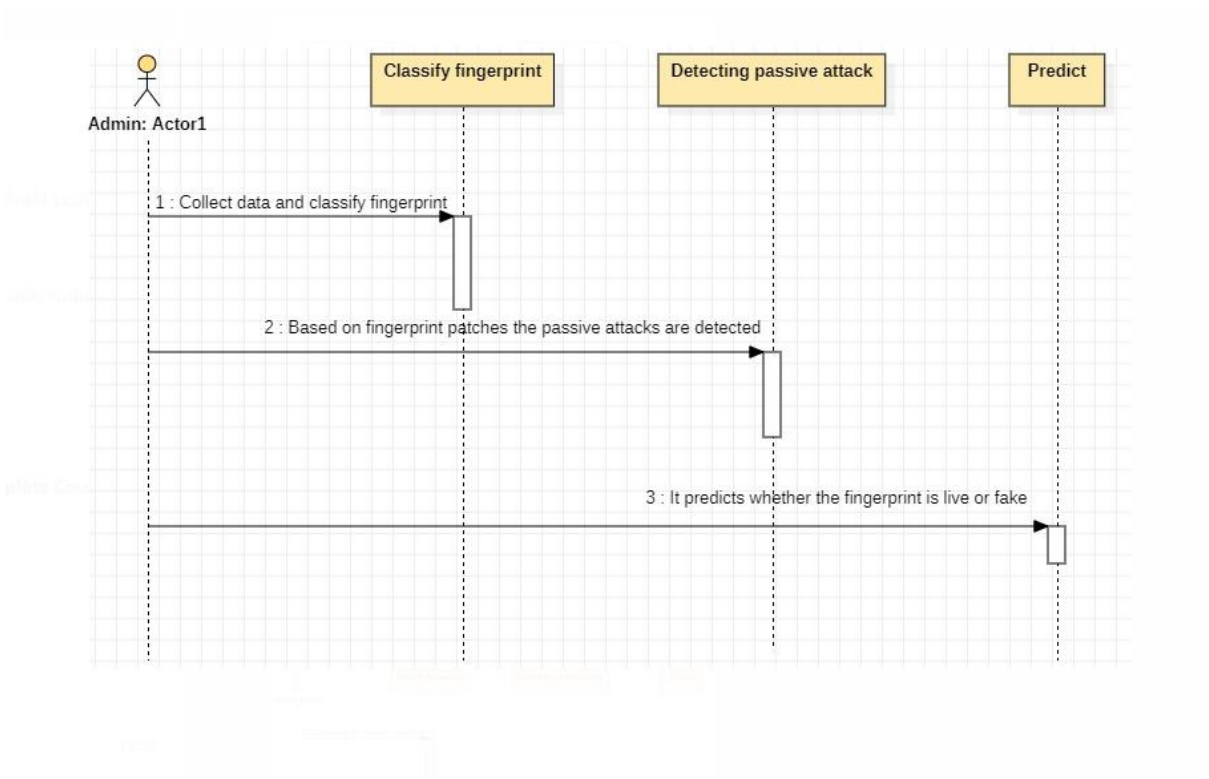
5.3 UML DIAGRAMS

5.3.1 USE CASE DIAGRAM



Figno:5.3 Use case Diagram

5.3.2SEQUENCE DIAGRAM

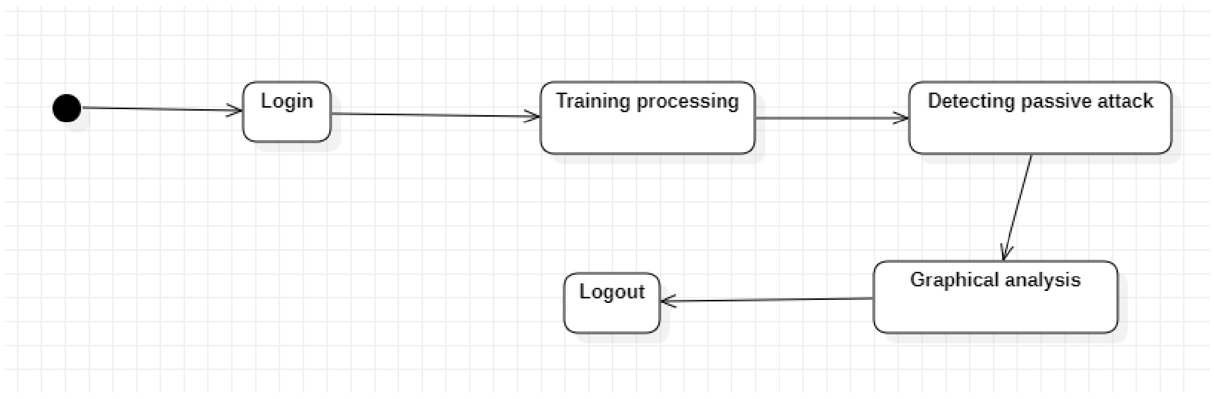


Admin: Actor1
Classify fingerprint
Detecting passive attack
Predict
1 : Collect data and classify fingerprint
2 : Based on fingerprint patches the passive attacks are detected
3 : It predicts whether the fingerprint is live or fake

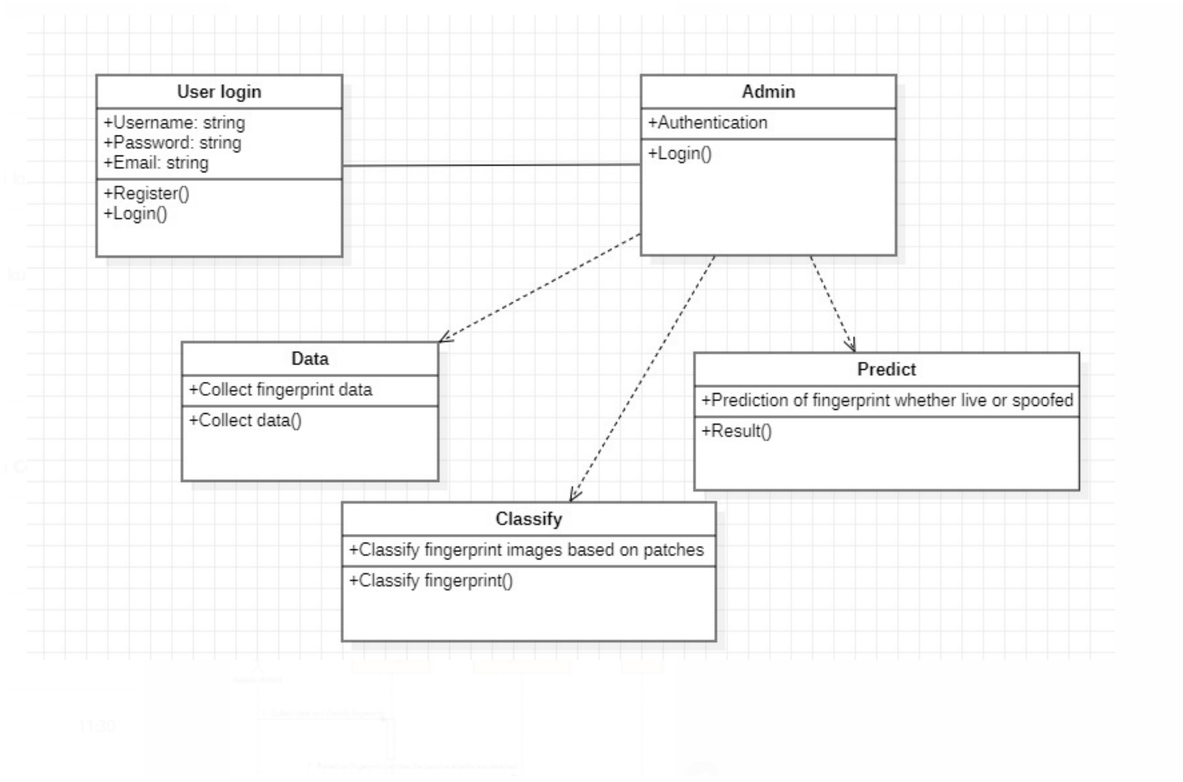
Figno:5.4 Architectural Diagram

5.3.3 ACTIVITY DIAGRAM

Figno:5.5 Activity Diagram



5.3.4 CLASS DIAGRAM



Figno:5.6 Class Diagram

CHAPTER 6

TESTING AND IMPLEMENTATION

6.1. VERIFICATION AND VALIDATION

The goals of verification and validation activities are to assess and improve the quality of the work products generated during development and modification of software. Quality attributes of interests include correctness, completeness, consistency, reliability, efficiency, conformance to standards, and overall cost effectiveness.

According to Boehm,

Verification: “Are we building the right product?”

Validation: “Are we building the right product?”

Verification and Validation involve assessment of work products to determine conformance to specifications. Specifications include the requirements specifications, the design documentation’s, various stylistic guidelines, implementation language standards, project standards, organizational standards and user expectations, as well as the meta-specifications for the formats and notations used in the various product specifications.

There are two types of verification:

1. Life-cycle verification and
2. Formal verification.

Life-cycle verification is the process determining the degree to which the work products of a given phase of the development cycle fulfill the specifications established during prior phases. Formal verification is a rigorous mathematical demonstration that source code conforms its specifications.

Validation is the process of evaluating software at the end of the software development process to determine compliance with the requirements.High quality cannot be achieved through testing of source code alone. Although a program should be totally free of errors, this is seldom the case for large software products. Even if source code errors were the only measure of quality, testing alone cannot guarantee the absence of errors in a program. A well-known maxim states that the number of bugs remaining in a program is proportional to the number already discovered. This is because one has most confidence in programs with no detected bugs after thorough testing and least confidence in a program with a long history of fixes.

The best way to minimize the number of errors in a program is to catch and remove the errors during analysis and design, so that few errors are introduced into the source code. Verification and validation are pervasive concepts, not a set of activities that occur strictly following implementation.

6.2. TESTING

Once the entire system has been built then it has to be tested against the "System Specification" to check if it delivers the features required. It is still developer focused, although specialist developers known as systems testers are normally employed to do it. In essence System Testing is not about checking the individual parts of the design, but about checking the system as a whole. In effect it is one giant component.

Testing presents an interesting challenge for the software programmer. In testing, the programmer creates a series of test cases that are intended to demolish the software that has been built. Testing requires that the developer discard preconceived notions of the correctness of the software developed and overcome a conflict of interest that occurs when errors are uncovered.

TESTING OBJECTIVES

There are several rules can serve as testing objectives. They are

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test is one that has a high probability of finding an undiscovered error.
3. A successful test is one that uncovers and undiscovered error.

The above are the objectives for a good test procedure. A successful test is that in which no errors are found. The objective is to design tests that systematically uncover different classes of errors and to do so with the minimum amount of time and effort.

6.3. TYPE OF TESTING

6.3.1. UNIT TESTING

Instead of testing the system as a whole, unit testing focuses of the modules that make up the system. Each module is taken up individually and tested for correctness in coding and login. Error resulting from interaction of modules is initially avoided.

1. Size of module is quite small that errors can easily are.
2. Located
3. Confusing interactions of multiple errors in wide
4. Different parts of the software eliminated
5. Modules level testing can be exhaustive.

6.3.2. INTEGRATION TESTING

It tests for the errors resulting from integration of modules. One specification of integration testing is the interface whether parameters match on both sides of type permissible ranges and meaning. Analysts try to find areas different specification for data elements. Integration testing is functional of black box testing method. That is using testing, each module is treats as an impenetrable mechanism for information a mechanism. The only concern during integration testing is that the modules work together properly.

6.3.3. VALIDATION TESTING

Validation test can be defined in many ways, but a simple definition is that validation succeeds when the software function is a manner that can be reasonably expected by the customer in this project, if we are type the name is not string give the error message type the name is only string.

6.3.4. OUTPUT TESTING

After performing the validation testing the next step is output testing. Here, the output format is considered in two ways. One is a screen and another is printed format. The output format on the screen is found to be correct as the format was designed in the system design phase according to the user need. For the hard copy also, the output testing has not resulted in many corrections in the system.

6.3.5. USER ACCEPTANCE TESTING

Acceptance Testing checks the system against the "Requirements". It is similar to systems testing in that the whole system is checked but the important difference is the change in focus. Systems Testing checks that the system that was specified has been delivered. Acceptance Testing checks that the system delivers what was requested. The customer, and not the developer should always do acceptance testing. The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgment.

6.3.6. WHITE BOX TESTING

The code testing strategy examines the logic of the program. To follow this testing method, the analyst develops test cases that result in executing every instruction in the program or module in every path through the program is tested. A path is a specific combination of conditions that is handled by the program. Code testing also does not check the range of data that the program will accept, even though, when software failures occursin actual use, it submitted data outside of expected ranges.

6.3.7. BLACK BOX TESTING

To perform specification testing, the analyst examine the specification starting what the program should do and how it should perform under various conditions. Then test cases are developed for each condition or combinations of conditions and submitted for processing. By examining the result, the analyst can determine whether the programs perform according to its specified requirements. This testing strategy on the face of it sounds exhaustive. If every statement in the program is checked for its validity, their doesn't seem to be much scope for errors.

6.4. SYSTEM IMPLEMENTATION

Implementation is the state in the system where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that will work efficiently and effectively. The system can be implemented only after thorough testing in done and if found to work according to the specification.

If involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of changeover methods apart from planning. Two major tasks of preparing the implementation are education, training of the users and testing the systems. System analysis and design efforts will be more for complex systems beings implemented. Based on policies of individual's organization an implementation coordinating committee has been appointed.

The implementation process begins with preparing a plan for the implementation system. According to this plan, the other activities are to be carried out. In this plan, discussion has been made regarding the equipment, resources and how to test the activities. Thus a clear plan is preparing for the activities.

The process of a biometric system detecting a biometric spoof is known as Presentation Attack Detection (PAD). PAD systems utilize a combination of hardware and software technologies to determine whether a biometric system is genuine. A liveness detection is a subset of presentation attack detection. All the data during spoof detection through PAD method are stored in LIVDET server. From LIVDET server genuine and forged fingerprint images are collected for training and testing the model.

During cross-validation the entire dataset will be divided into two sets, the training set and testing set. 80% of the dataset goes to the training set and 20% to the test set. The model will be trained on the training set and tested on the test set.

Using softmax classifier the fingerprints were classified as genuine or forged. Using collected finger print images from LiveDet, apply the CNN algorithm to predict the result and identify the finger print whether it is live or spoofing. A combination of datasets had used in the training model which help to improve the accuracy.

CNN converts the fingerprint images into fingerprint patches. By analyzing the patches it decides whether it is spoofed or live fingerprint through more amount of training in the training process. As CNN converts fingerprint images into patches, it can predict the accurate result.

CHAPTER 7

APPENDIX

SAMPLE CODING

User/views.py

```
from django.shortcuts import render,redirect
from django.db.models import Avg, Count
from django.views.decorators.csrf import csrf_exempt
from django.http import JsonResponse
import numpy as np
import urllib
import json
import os
import requests
from tkinter import *
from PIL import Image
from PIL import ImageTk
from tkinter import filedialog
import tkinter.filedialog
import cv2
from skimage import exposure
import pickle
import sys
import imutils
from matplotlib import pyplot as plt
from skimage.measure import compare_ssim
from scipy import linalg
import numpy
import argparse
import numpy as np
import pandas as pd
import os
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import matplotlib.cbook
from django.http import HttpResponse
from django.shortcuts import render, redirect, get_object_or_404
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
import matplotlib.image as mpimg
import numpy as np
import matplotlib.pyplot as plt
import cv2
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
# Create your views here.
from user.models import userreg, fingerprint, accuracymodel
def user_index(request):
    return render(request,'user/user_index.html')
```

```
def user_login(request):
    if request.method == "POST":
uname = request.POST.get('uname')
pswd = request.POST.get('password')
        try:
            check = userreg.objects.get(uname=uname, password=pswd)
            print(check)
request.session['userid'] = check.id
request.session['username'] = check.uname
            return redirect('myaccounts')
        except:
            pass
        return redirect('user_login')
    return render(request,'user/user_login.html')
def user_register(request):
    if request.method == "POST":
        name = request.POST.get('name')
        email = request.POST.get('email')
        mobile = request.POST.get('mobile')
        gender = request.POST.get('gender')
        location = request.POST.get('location')
uname = request.POST.get('uname')
        password = request.POST.get('password')
userreg.objects.create(name=name,            email=email,            mobile=mobile,
gender=gender,
        location=location, uname=uname, password=password)
        return redirect('user_login')
    return render(request,'user/user_register.html')
def user_home(request):
    if request.method == "POST" and request.FILES['image']:
        image = request.FILES['image']
fingerprint.objects.create(fingerprintimage=image)
    return render(request,'user/user_home.html')
def myaccounts(request):
userd_id = request.session['userid']
    obj = userreg.objects.get(id=userd_id)
    if request.method == "POST":
        name = request.POST.get('name')
        mobile = request.POST.get('mobile')
        gender = request.POST.get('gender')
        password = request.POST.get('password')
        location = request.POST.get('location')
        email = request.POST.get('email')
uname = request.POST.get('uname')
        obj = get_object_or_404(userreg, id=userd_id)
        obj.name = name
obj.mobile = mobile
obj.gender = gender
obj.password = password
obj.location = location
obj.email = email
obj.uname = uname
obj.save(update_fields=["name","mobile",            "uname",        "password",
"location","email","gender"])
        return redirect('myaccounts')
    return render(request, 'user/myaccounts.html',{'form':obj})
def charts(request,chart_type):
    chart                                                                                    =
fingerprint.objects.values('analysisvalue').annotate(dcount=Count('analysisvalue
'))
    return render(request,'user/charts.html',{'chart_type':chart_type,'form':chart})
def accuracys(request):
    import os
    import cv2
    import numpy as np
    from tqdm import tqdm
    from skimage import feature
    from sklearn.svm import LinearSVC
    from sklearn.utils import shuffle
    from sklearn.metrics import confusion_matrix
    def main():
train_x = []
train_y = []
test_x = []
test_y = []
print("Extracting LBPH (Local Binary Pattern Histogram) features from the
training images (live)")
images_live = sorted(os.listdir("data/train_live"))
        for file in tqdm(images_live):
            file = "data/train_live/" + str(file)
            image = cv2.imread(file)
            cv2.resize(image, (64, 64))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            # Using Local Binary Pattern (LBP) for texture classification
lbp = feature.local_binary_pattern(image, 24, 8, method="uniform")
            # Getting histogram of LBP
            hist, bins = np.histogram(lbp.ravel(), bins=np.arange(0, 27), range=(0,
10))
            # normalize the histogram
            hist = hist.astype("float")
            hist /= (hist.sum() + 1e-7)
            # Adding Data Point TO Array
train_y.append(1)  # Live Samples Are The Positive Ones
train_x.append(hist)
print("Extracting LBPH (Local Binary Pattern Histogram) features from the
training images (spoof)")
images_spoof = sorted(os.listdir("data/train_spoof"))
        for file in tqdm(images_spoof):
            file = "data/train_spoof/" + str(file)
            image = cv2.imread(file)
            cv2.resize(image, (64, 64))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            # Using Local Binary Pattern (LBP) for texture classification
lbp = feature.local_binary_pattern(image, 24, 8, method="uniform")
            # Getting histogram of LBP
            hist, bins = np.histogram(lbp.ravel(), bins=np.arange(0, 27), range=(0,
10))
            # normalize the histogram
            hist = hist.astype("float")
            hist /= (hist.sum() + 1e-7)
            # Adding Data Point TO Array
train_y.append(0)  # Spoof Samples Are The Negative Ones
train_x.append(hist)
train_x, train_y = shuffle(train_x, train_y)
```

```
print("Extracting LBPH (Local Binary Pattern Histogram) features from the
testing images (live)")
images_live = sorted(os.listdir("data/test_live"))
        for file in tqdm(images_live):
            file = "data/test_live/" + str(file)
            image = cv2.imread(file)
            cv2.resize(image, (64, 64))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            # Using Local Binary Pattern (LBP) for texture classification
lbp = feature.local_binary_pattern(image, 24, 8, method="uniform")
            # Getting histogram of LBP
            hist, bins = np.histogram(lbp.ravel(), bins=np.arange(0, 27), range=(0,
10))
            # normalize the histogram
            hist = hist.astype("float")
            hist /= (hist.sum() + 1e-7)

            # Adding Data Point TO Array
test_y.append(1)  # Live Samples Are The Positive Ones
test_x.append(hist)
print("Extracting LBPH (Local Binary Pattern Histogram) features from the
testing images (spoof)")
images_spoof = sorted(os.listdir("data/test_spoof"))
        for file in tqdm(images_spoof):
            file = "data/test_spoof/" + str(file)
            image = cv2.imread(file)
            cv2.resize(image, (64, 64))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            # Using Local Binary Pattern (LBP) for texture classification
```

```
lbp = feature.local_binary_pattern(image, 24, 8, method="uniform")
            # Getting histogram of LBP
            hist, bins = np.histogram(lbp.ravel(), bins=np.arange(0, 27), range=(0,
10))
            # normalize the histogram
            hist = hist.astype("float")
            hist /= (hist.sum() + 1e-7)
            # Adding Data Point TO Array
test_y.append(0)  # Spoof Samples Are The Negative Ones
test_x.append(hist)
train_x, train_y = shuffle(train_x, train_y)
test_x, test_y = shuffle(test_x, test_y)
print("Training CNN Model..")
        # Train SVM on the data
        model = LinearSVC(C=100, max_iter=10000)
model.fit(train_x, train_y)
print("Training Completed!\n\nTesting Now...")
        pred = model.predict(test_x)
con_matrix = confusion_matrix(test_y, pred)  # ,labels=["Live","Fake"])
        TP = con_matrix[0][0]
        FN = con_matrix[0][1]
        FP = con_matrix[1][0]
        TN = con_matrix[1][1]
print("Precision of the CNN:", round((TP / (TP + FP)), 3))
        pre=round((TP / (TP + FP)), 3)
        print(pre)
print("Recall of the CNN:", round((TP / (TP + FN)), 3))
        re = round((TP / (TP + FN)), 3)
        print(re)
print("Accuracy of the CNN:", round(((TP + TN) / (TP + TN + FP + FN)), 3))
        acc = round((TP + TN) / (TP + TN + FP + FN), 3)
        print(acc)
accuracymodel.objects.create(precision=pre,recall=re,accuracy=acc)

print("Fingerprint spoof detection system based on two-class CNN")
main()
Vobj = accuracymodel.objects.order_by('-id')[:1]
    return render(request,'user/accuracys.html', {'v': Vobj})
def fprint(request):
ind=''
    def select_image1():
        # grab a reference to the image panels
        global panelA, panelB
        # open a file chooser dialog and allow the user to select an input
        # image
        path = filedialog.askopenfilename()
        print(path)
        # Initialising the CNN
        classifier = Sequential()
        # Convolution
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
        # Pooling
classifier.add(MaxPooling2D(pool_size=(2, 2)))
        # Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
        # Flattening
classifier.add(Flatten())
```

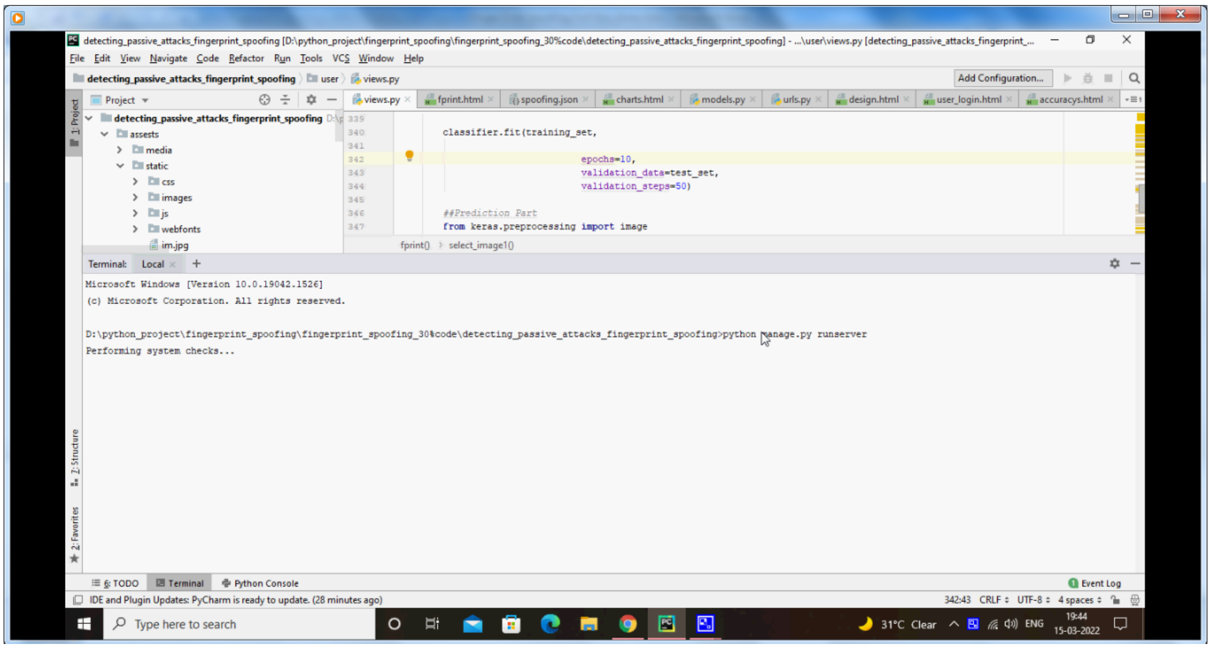
```
        # Full connection
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))
        # Compiling the CNN
classifier.compile(optimizer='adam',                          loss='binary_crossentropy',
metrics=['accuracy'])
        # Fitting the CNN to the images
train_datagen = ImageDataGenerator(rescale=1. / 255,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1. / 255)
training_set = train_datagen.flow_from_directory('dataset/training',
target_size=(64, 64),
batch_size=32,
class_mode='binary')

test_set = test_datagen.flow_from_directory('dataset/testing',
target_size=(64, 64),
batch_size=32,
class_mode='binary')
classifier.fit_generator(training_set,
steps_per_epoch=100,
                                epochs=10,
validation_data=test_set,
validation_steps=50)
        ##Prediction Part
        from keras.preprocessing import image
img_pred = image.load_img(path, target_size=(64, 64))
img_pred = image.img_to_array(img_pred)
img_pred = np.expand_dims(img_pred, axis=0)
rslt = classifier.predict(img_pred)
ind = training_set.class_indices
print()
        if rslt[0][0] == 1:
            prediction = "Live"
        else:
            prediction = "Fake"
        image = cv2.imread(path)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        ted = gray
        ret, thresh1 = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
        (score, diff) = compare_ssim(ted, thresh1, full=True)
        diff = (diff * 255).astype("uint8")
        ret, thresh1 = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
        (score, diff) = compare_ssim(ted, thresh1, full=True)
        diff = (diff * 255).astype("uint8")
clssf = classifier.to_json()
        with open("spoofing.json", "w") as json_file:
json_file.write(clssf)
classifier.save_weights("spoofing.h5")
print("model saved to disk....")
        print(prediction)
ind = training_set.class_indices
        if rslt[0][0] == 1:
            prediction = "Live"
        else:
            prediction = "Fake"
```

```
target_width = 50
target_height = 50
target_size = (target_width, target_height)
        thresh = cv2.threshold(diff, 0, 255,
                        cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
img_src = path
img = cv2.imread(img_src)
        ret, thresh = cv2.threshold(thresh1, 0, 255, cv2.THRESH_BINARY_INV
+ cv2.THRESH_OTSU)
dst = cv2.GaussianBlur(thresh1, (5, 5), cv2.BORDER_DEFAULT)
        edges = cv2.Canny(thresh1, 100, 200)
        kernel = np.ones((5, 5), np.uint8)
img_erosion = cv2.erode(thresh1, kernel, iterations=1)
        edged = cv2.Canny(gray, 50, 100)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(image)
        edged = Image.fromarray(edged)
        image = ImageTk.PhotoImage(image)
        edged = ImageTk.PhotoImage(edged)
cnts = cv2.findContours(img_erosion.copy(), cv2.RETR_EXTERNAL,
                cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
        print("[INFO] {} unique contours found".format(len(cnts)))
uniquecon = ("{}".format(len(cnts)))
        # loop over the contours
img = cv2.resize(img, target_size)
img = img.reshape(1, target_width, target_height, 3
        fig, ax = plt.subplots()
fig.suptitle(prediction, fontsize=12)
np_img = mpimg.imread(img_src)
        cv2.imshow(' Grayscale Image', ted)
        cv2.imshow('Binary Threshold', thresh1)
plt.subplot(121), plt.imshow(dst, cmap='gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(edges, cmap='gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.imshow(np_img)
plt.show()
        name = request.session['userid']
userObj = userreg.objects.get(id=name)
fingerprint.objects.create(userd=userObj,                   analysisvalue=prediction,
fingerprintimage=path)
    # initialize the window toolkit along with the two image panels
    root = Tk()
panelA = None
panelB = None
    # create a button, then when pressed, will trigger a file chooser
    # dialog and allow the user to select an input image; then add the
    # button the GUI
btn =Button(root, text="Select Fingerprint Image", command=select_image1)
btn.pack(side="bottom", fill="both", expand="yes", padx="10", pady="10")
root.mainloop()
Vobj = fingerprint.objects.order_by('-id')[:1]
    return render(request, 'user/fprint.html', {'v': Vobj})
```

CHAPTER 8

OUTPUT SCREENSHOTS



```
classifier.fit(training_set,

                epochs=10,
                validation_data=test_set,
                validation_steps=50)

##Prediction Part
from keras.preprocessing import image
```

```
Microsoft Windows [Version 10.0.19042.1526]
(c) Microsoft Corporation. All rights reserved.

D:\python_project\fingerprint_spoofing\fingerprint_spoofing_30%code\detecting_passive_attacks_fingerprint_spoofing>python manage.py runserver
Performing system checks...
```

Fig no:8.1 Command to run the project

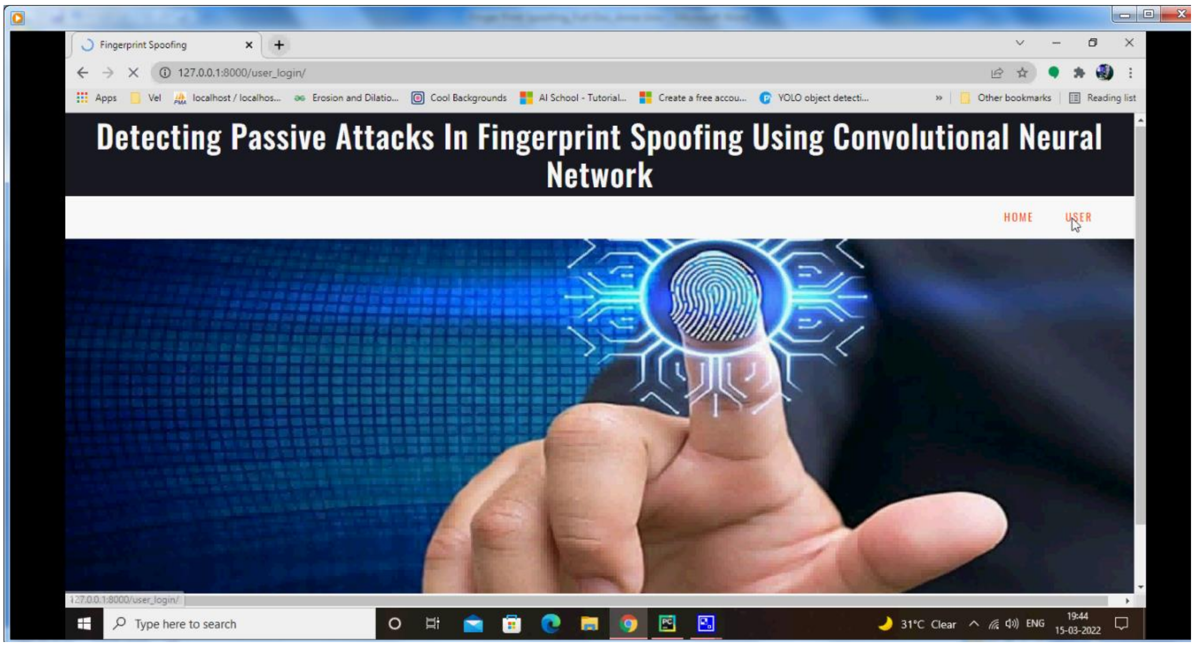

Fingerprint Spoofing
127.0.0.1:8000/user_login/
Detecting Passive Attacks In Fingerprint Spoofing Using Convolutional Neural Network
HOME
USER

Fig no:8.2 Homepage

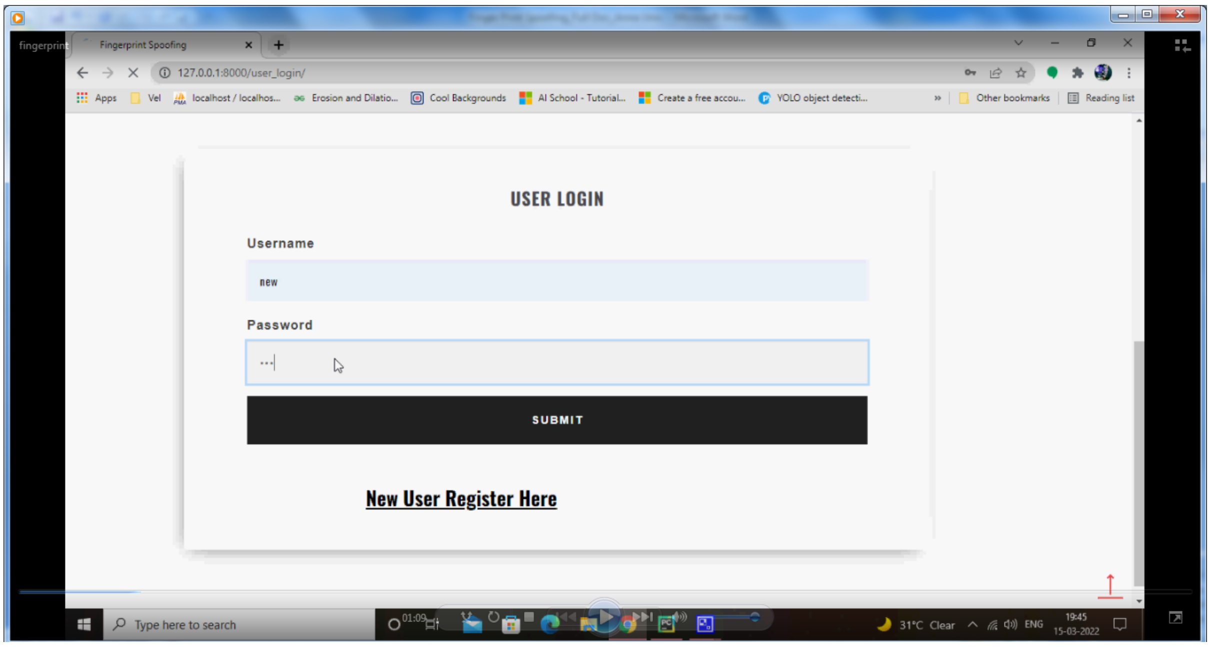
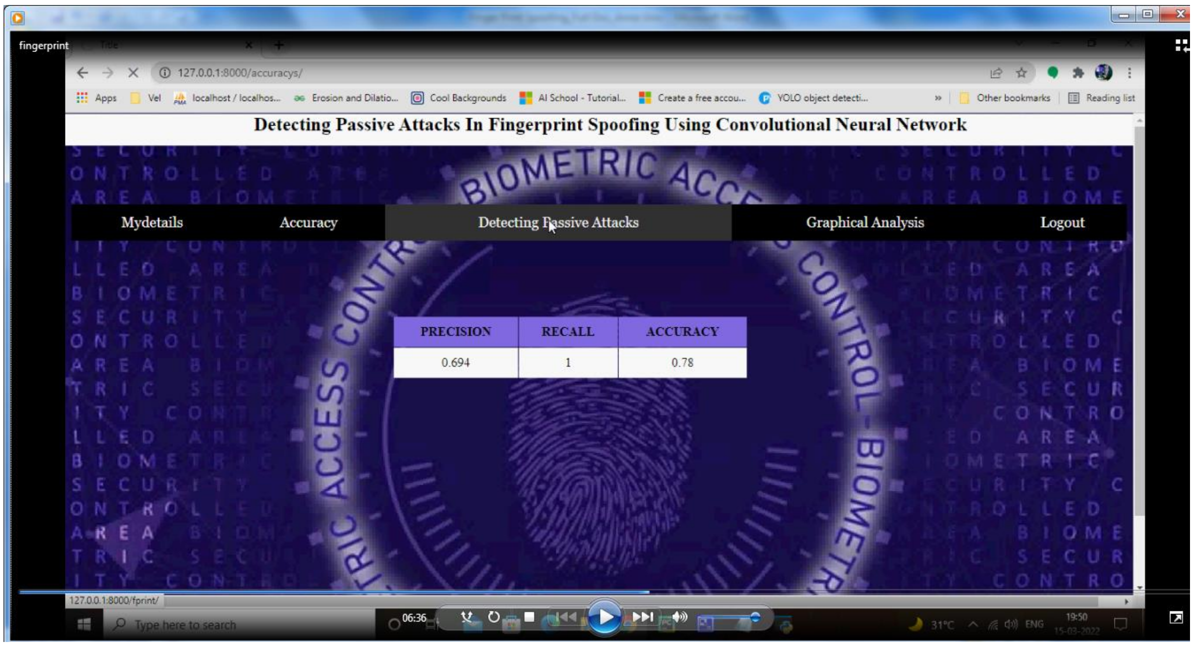


Fig no:8.3 User Login


Detecting Passive Attacks In Fingerprint Spoofing Using Convolutional Neural Network
Mydetails
Accuracy
Detecting Passive Attacks
Graphical Analysis
Logout

PRECISION	RECALL	ACCURACY
0.694	1	0.78

Fig no:8.4 Accuracy of Training Process

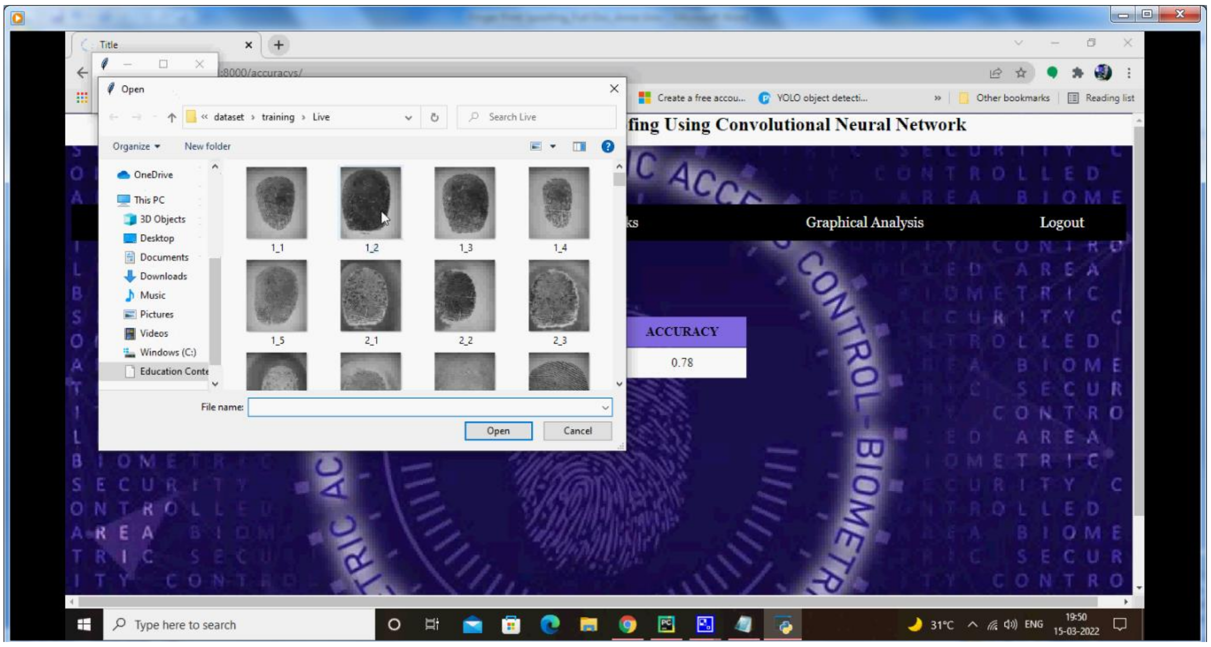
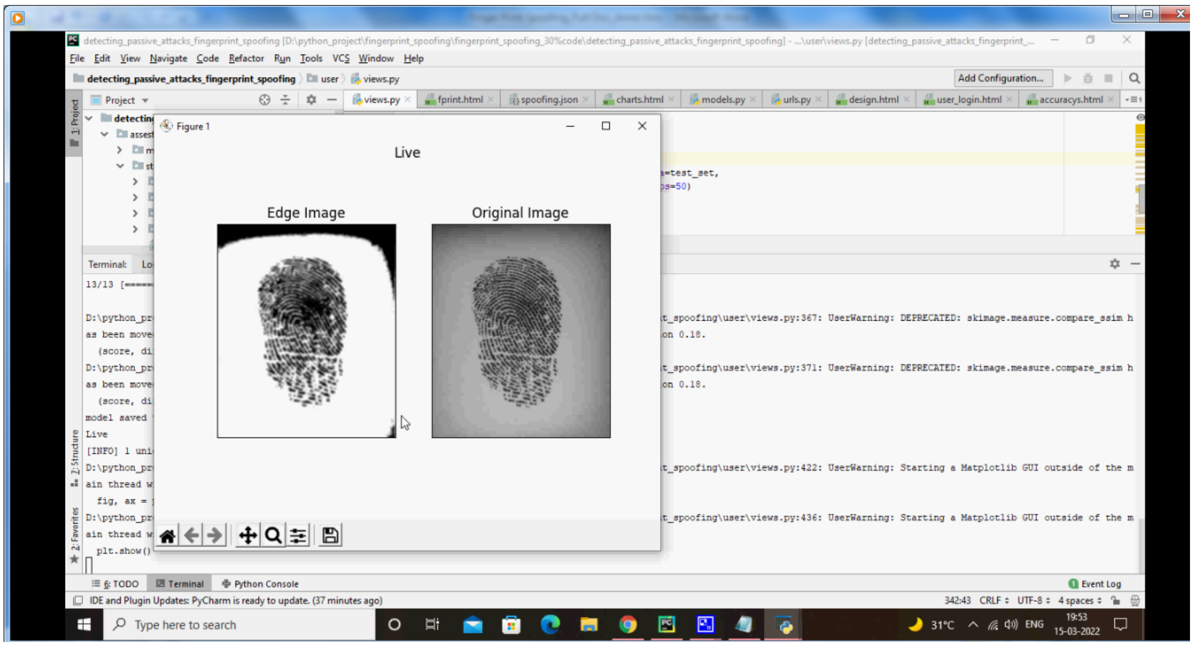


Fig no:8.5 Select Finger print image to detect passive attack

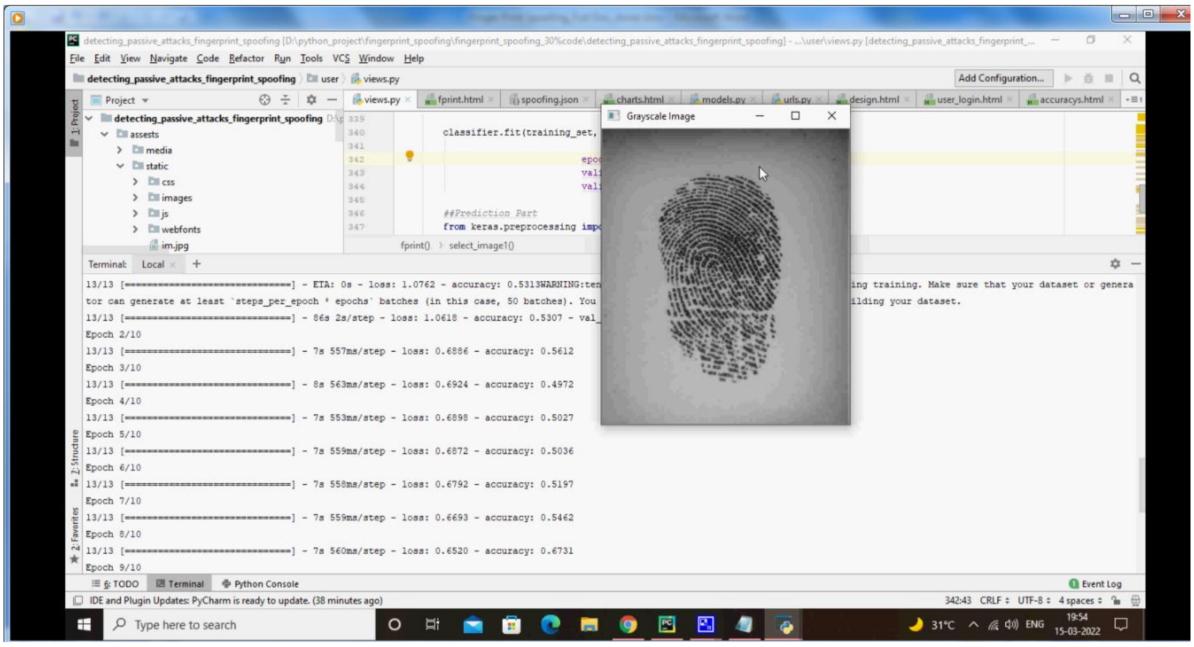

detecting_passive_attacks_fingerprint_spoofing [D:\python_project\fingerprint_spoofing\fingerprint_spoofing_30%code\detecting_passive_attacks_fingerprint_spoofing] - ...\user\views.py [detecting_passive_attacks_fingerprint_...
File
Edit
View
Navigate
Code
Refactor
Run
Tools
VCS
Window
Help
detecting_passive_attacks_fingerprint_spoofing
user
views.py
Add Configuration...
Project
views.py
fprint.html
spoofing.json
charts.html
models.py
urls.py
design.html
user_login.html
accuracys.html
Figure 1
Live
Edge Image
Original Image
Terminal:
6: TODO
Terminal
Python Console
Event Log
IDE and Plugin Updates: PyCharm is ready to update. (37 minutes ago)
342:43 CRLF UTF-8 4 spaces
Type here to search
31°C
ENG
19:53
15-03-2022

```
=test_set,
s=50)
```

```
13/13 [=====

D:\python_pr                t_spoofing\user\views.py:367: UserWarning: DEPRECATED: skimage.measure.compare_ssim h
as been move                on 0.18.
  (score, di
D:\python_pr                t_spoofing\user\views.py:371: UserWarning: DEPRECATED: skimage.measure.compare_ssim h
as been move                on 0.18.
  (score, di
model saved
Live
[INFO] 1 uni
D:\python_pr                t_spoofing\user\views.py:422: UserWarning: Starting a Matplotlib GUI outside of the m
ain thread w
  fig, ax =
D:\python_pr                t_spoofing\user\views.py:436: UserWarning: Starting a Matplotlib GUI outside of the m
ain thread w
  plt.show()
```

Fig no:8.6 Identifying fingerprint (Live)

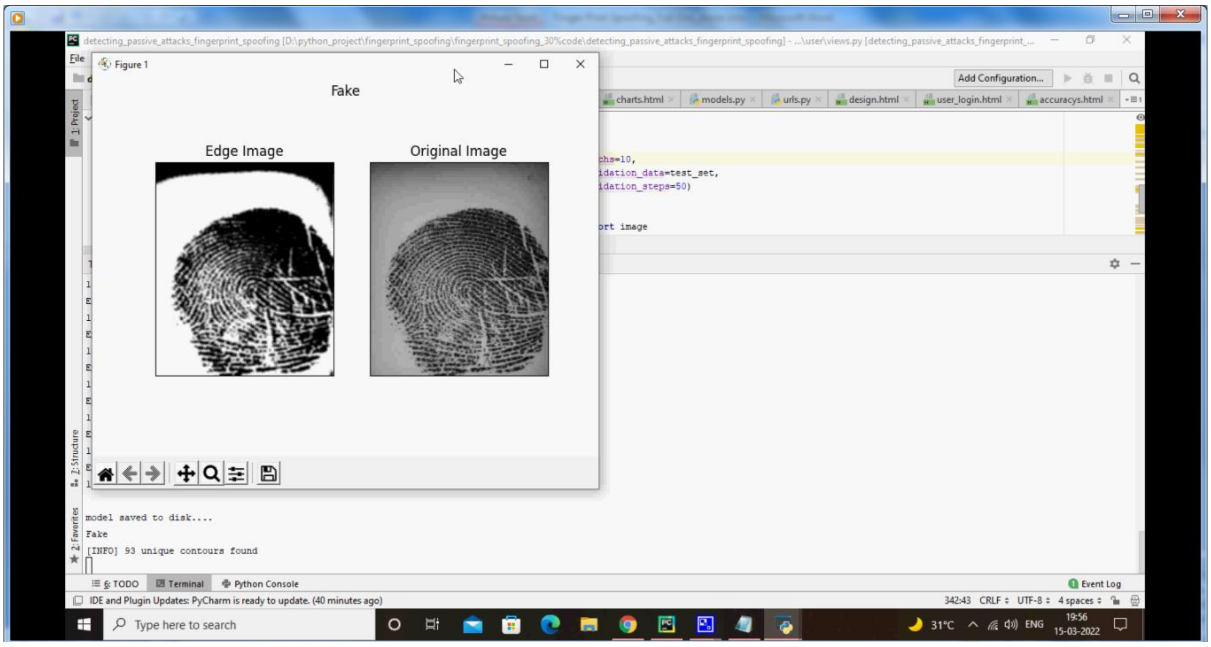


```
classifier.fit(training_set,

##Prediction Part
from keras.preprocessing impo
```

```
13/13 [==============================] - ETA: 0s - loss: 1.0762 - accuracy: 0.5313WARNING:ten        ing training. Make sure that your dataset or genera
tor can generate at least `steps_per_epoch * epochs` batches (in this case, 50 batches). You        ilding your dataset.
13/13 [==============================] - 86s 2s/step - loss: 1.0618 - accuracy: 0.5307 - val_
Epoch 2/10
13/13 [==============================] - 7s 557ms/step - loss: 0.6886 - accuracy: 0.5612
Epoch 3/10
13/13 [==============================] - 8s 563ms/step - loss: 0.6924 - accuracy: 0.4972
Epoch 4/10
13/13 [==============================] - 7s 553ms/step - loss: 0.6898 - accuracy: 0.5027
Epoch 5/10
13/13 [==============================] - 7s 559ms/step - loss: 0.6872 - accuracy: 0.5036
Epoch 6/10
13/13 [==============================] - 7s 558ms/step - loss: 0.6792 - accuracy: 0.5197
Epoch 7/10
13/13 [==============================] - 7s 559ms/step - loss: 0.6693 - accuracy: 0.5462
Epoch 8/10
13/13 [==============================] - 7s 560ms/step - loss: 0.6520 - accuracy: 0.6731
Epoch 9/10
```

Fig no:8.7 Gray Scale Image



```
chs=10,
idation_data=test_set,
idation_steps=50)

ort image
```

```
model saved to disk....
Fake
[INFO] 93 unique contours found
```

Fig no:8.8 Identifying fingerprint (Fake)

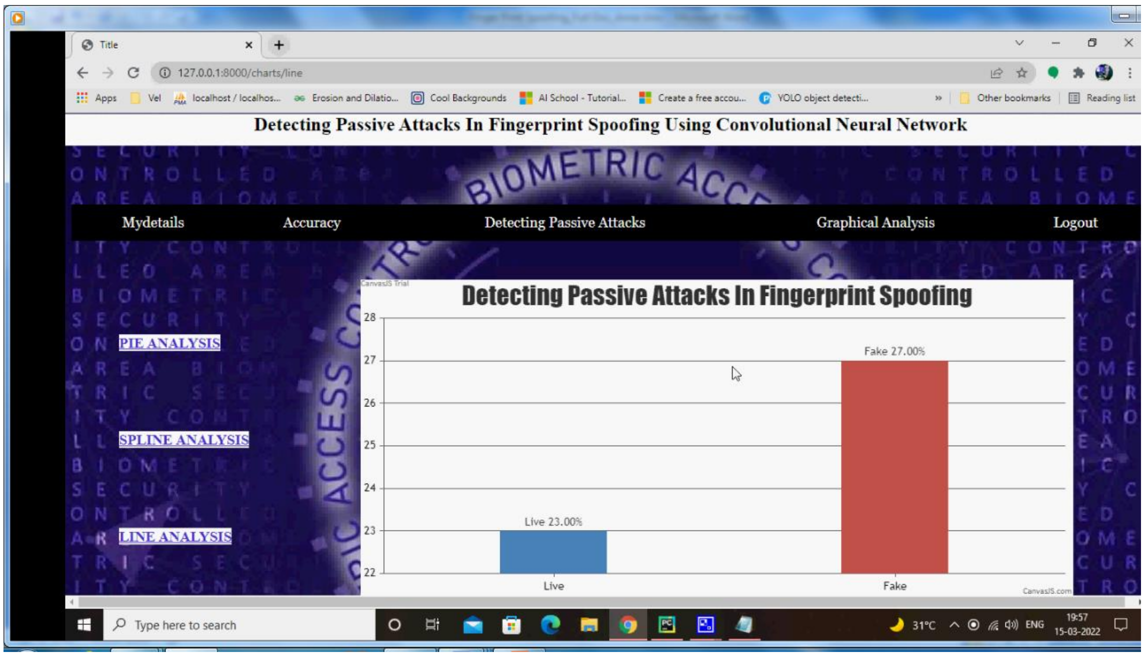

Detecting Passive Attacks In Fingerprint Spoofing Using Convolutional Neural Network
Mydetails
Accuracy
Detecting Passive Attacks
Graphical Analysis
Logout
PIE ANALYSIS
SPLINE ANALYSIS
LINE ANALYSIS
Detecting Passive Attacks In Fingerprint Spoofing
Live 23.00%
Fake 27.00%
Live
Fake

Fig no:8.9 Analysis1 (Pie Chart)

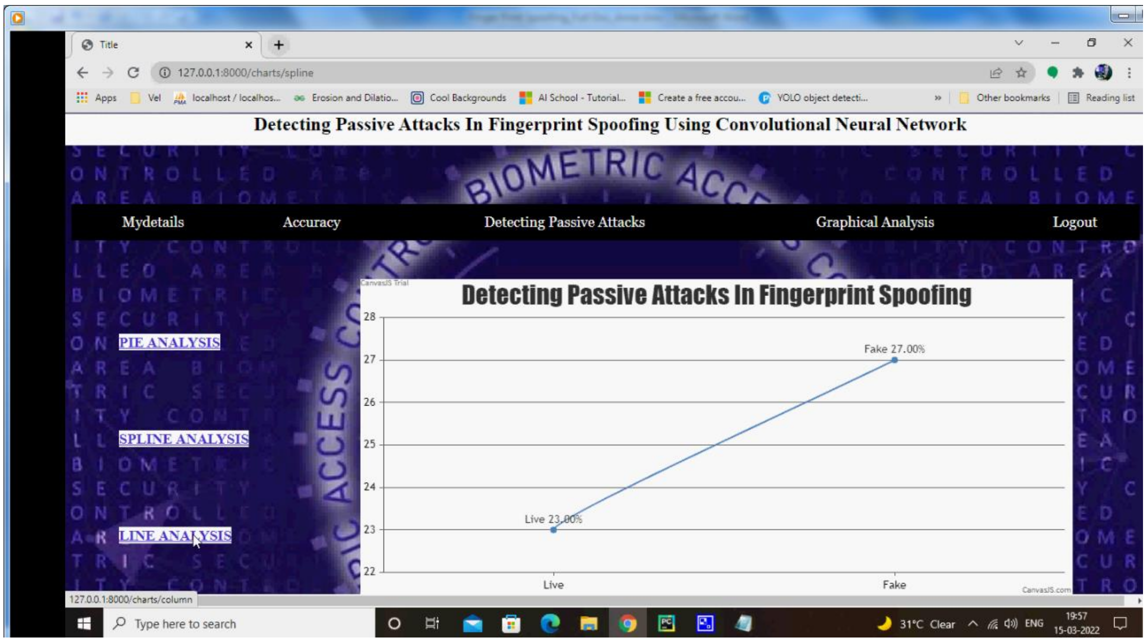

Detecting Passive Attacks In Fingerprint Spoofing Using Convolutional Neural Network
Mydetails
Accuracy
Detecting Passive Attacks
Graphical Analysis
Logout
PIE ANALYSIS
SPLINE ANALYSIS
LINE ANALYSIS
Detecting Passive Attacks In Fingerprint Spoofing
Live 23.00%
Fake 27.00%
Live
Fake

Fig no:8.10 Analysis2 (S Line Chart)

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENTS

9.1. CONCLUSION

This novel approach, which considers a scheme that can be extended as an architecture for liveness-server for extraction/updating transient liveness factor. The liveness identification is penultimate scope during design, development for testifying the FPAD system to a user. Efforts can used to model a system with generalization. We have discussed here a small step forward towards generalized and reproducible system. This addresses a concern towards the requirement for the uniform system is need of today for future application. The approach proposed will act as a technology of future. A wider scope (usages) over smartphones and smart devices apart from conventional devices is possible then. Only requirement is to design a liveness server according to end user machine(s). With this intelligence gained from TLF, the liveness server will become a new biometric technology backbone. Hackers will find it difficult to access any device. Because of this dynamism, the future machine may match the user access pattern together for more robust system.

9.2. FUTURE ENHANCEMENT

Using collected finger print images from LiveDet, we apply the CNN algorithm to predict the result and identify the finger print whether it is live or spoofing. A combination of datasets had used in the training model which help to improve the accuracy. In future, we implement and develop this concept using new, primitive and innovative technology algorithm to provide more accuracy and more efficiency. We can also implement it speed process of execution to become robust identification.

CHAPTER 10

REFERENCES

[1] Rutba Ishfaq, Arvind Selwal, Deepika Sharma,” Fingerprint spoofing attack & their Deep Learning-enabled Remediation: State of the art, Taxonomy, and Future Directions DOI 10. 1109/TCCN.2021 Journal: IEEE Access.

[2] Title: "Deep learning-based spoof fingerprint detection using convolutional neural networks" Authors: Liu, C., Li, Y., Li, L., & Yang, J. Journal: IEEE Access.

[3] X. Guo, F. Wu, and X. Tang, "Fingerprint Pattern Identification And Classification", 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), pp. 1045-1050, 2018.

[4] M. Gomez-Barrero, R. Tolosana, R. Vera-Rodriguez, J. Fierrez, and R. Chaves, "LivDet-Iris 2017: A comparative evaluation of state-of-the-art iris presentation attack detection", IEEE Transactions on Information Forensics and Security, vol. 14, no. 9, pp. 2344-2357, Sep. 2019.

[5] S. R. Borra, G. J. Reddy, E. S. Reddy, "Classification of fingerprint images with the aid of morphological operation and AGNN classifier", Applied Computing and Informatics, Vol.14, No.2, pp.166-176, 2018.

[6] R. P. Krish et al., "Improving automated latent fingerprint identification using extended minutia types", Information Fusion, Vol.50, pp.9-19, 2020.

[7] S. Khade, S. D. Thepade, and A. Ambedkar, "Fingerprint Liveness Detection Using Directional Ridge Frequency with Machine Learning Classifiers", in 2018 Fourth International Conference on Computing Communication Contro and Automation (ICCUBEA).pp.1-5, 2020.

[8] Q. Huang, S. Chang, C. Liu, B. Niu, M. Tang, and Z. Zhou, "An evaluation of fake fingerprint datasets utilizing SVM classification", Patt. Recogn. Lett., pp.1-7, 2019.