TEXTURE AND QUALITY ANALYSIS FOR FACE SPOOFING DETECTION USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES

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

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

G.Indhumathi (20PIT004)

Under the Guidance of

Mrs.S.Karthika

M.C.A., M.Phil,NET.

Assistant Professor

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

TEXTURE AND QUALITY ANALYSIS FOR FACE SPOOFING DETECTION USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES

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

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

G.Indhumathi (20PIT004)

Under the Guidance of

Mrs.S.Karthika

M.C.A., M.Phil,NET.

Assistant Professor

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

DECLARATION

DECLARATION

I hereby declare that the project entitled "**TEXTURE AND QUALITY ANALYSIS FOR FACE SPOOFING DETECTION USING ML AND DL TECHNIQUES"** is a record Of the original work done by G.Indhumathi (20PIT004) under the guidance of **Mrs.S.Karthika M.C.A.,M.Phil,NET.** Assistant Professor, Department of Information Technology, School of Physical Sciences and Computational Sciences**,** Avinashilingam Institute for Home Science and Higher Education for Women in the partial fulfilment 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:

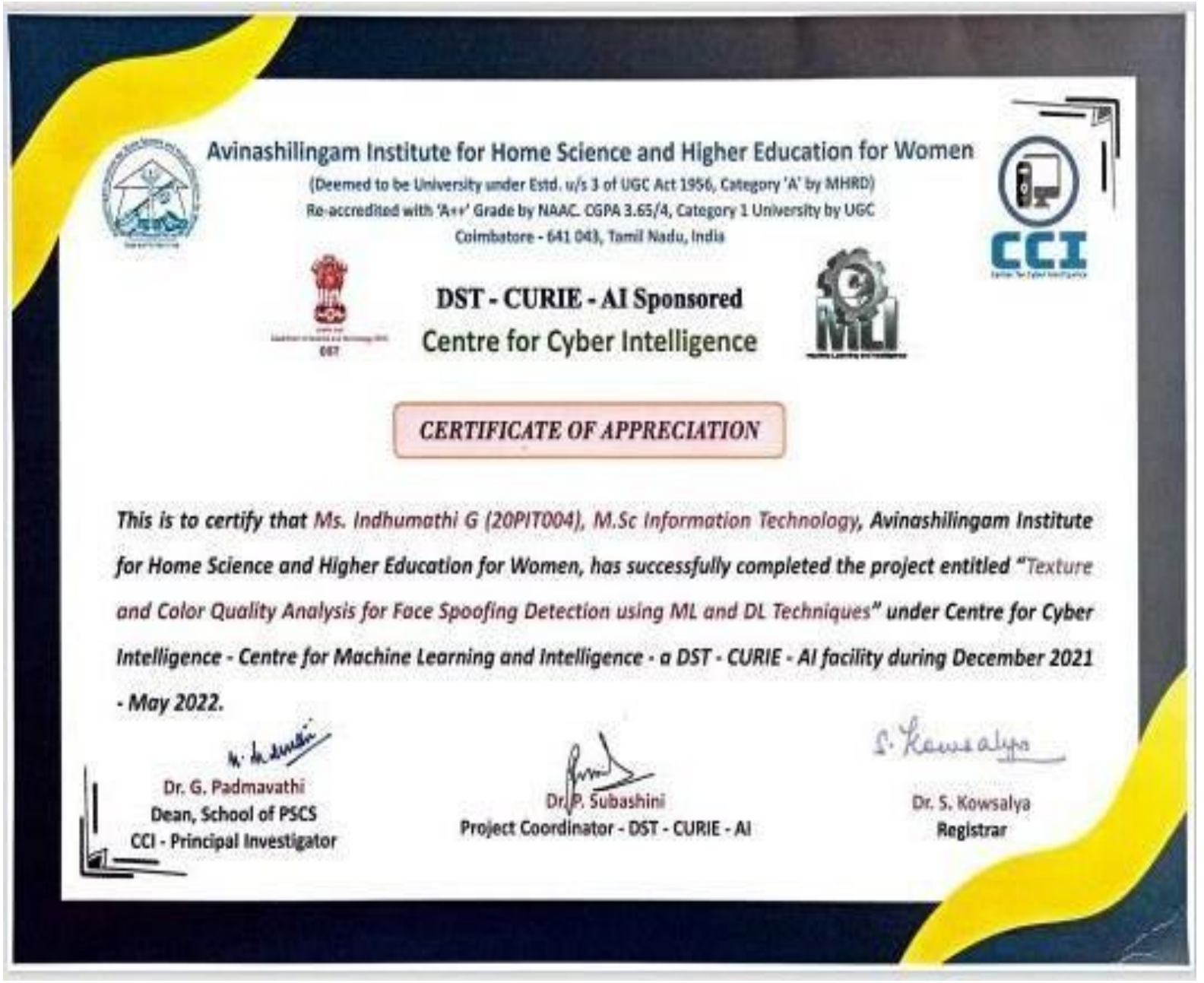
Signature of the Candidate

Countersigned By

Mrs.S.Karthika M.C.A., M.Phil, NET.

Assistant Professor,
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. Indhumathi G (20PIT004), M.Sc Information Technology, Avinashilingam Institute for Home Science and Higher Education for Women, has successfully completed the project entitled "Texture and Color Quality Analysis for Face Spoofing Detection using ML and DL Techniques" 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 "**TEXTURE AND QUALITY ANALYSIS FOR FACE SPOOFING DETECTION USING ML AND DL TECHNIQUES"** done by G.Indhumathi (20PIT004) has been submitted to Avinashilingam Institute for Home science and Higher education for women, Coimbatore-43 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 bonafied record of the work submitted for the Viva voce held on _____.

Signature of the HOD **Signature of the Guide**

Signature of External Examiner

ACKNOWLEGEMENT

ACKNOWLEDGEMENT

I take this opportunity to express our deep sense of gratitude to our **Chancellor Dr. S. P. Thyagarajan,** Avinashilingam Institute of Home Science and Higher Educationfor women, Coimbatore for his support and encouragement during the course of our project work.

I heartily thank **Dr. (Mrs.) Bharathi Harishankar Ph.D., FRSA**. **Vice-chancellor** for extending all resources that facilitated the conduct of the present work.

I wish to extend our sincere thanks to **Dr. (Mrs) S.Kowsalya** M.Sc, M.Phil, and Ph.D. **Registrar** for helping and sustaining me in all possible means to come outwith the project.

I wish to place on record our deep sense of gratitude to **Dr. (Mrs.) G.Padmavathi** M.Sc., M.Phil., Ph.D., **Dean**, School of Physical Sciences and Computational Sciences, for providing all the facilities to complete the project.

Our heartiest thanks to **Dr. (Mrs.) D.Shanmugapriya** M.Sc., M.Phil., Ph.D., SET **Head of Department** of Information Technology for the valuable guidance and encouragement duringthe course of our project.

I take this opportunity to express our profound gratitude and deep regards to our **Guide Mrs.S.Karthika M.C.A, M.phil.** For her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry us a long way in the journey of life on which we are about to embark.

I would like to thank all the faculty member laboratory staff of the Department of Information Technology, all our friends and well- wishers who had either directly or indirectly helped us to finish this Project successfully. I thank our parents for their encouragement and moral support. Above all the thank god almighty whose grace was sufficient at all times.

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

ABSTRACT

ABSTRACT

Current face biometric systems are vulnerable to spoofing attacks. A spoofing attack occurs when a person tries to masquerade as someone else by falsifying data and thereby gaining illegitimate access. Inspired by image quality assessment, characterization of printing artifacts, and differences in light reflection, I propose to approach the problem of spoofing detection from texture analysis point of view. Indeed, face prints usually contain printing quality defects that can be well detected using texture and local shape features.

The proposed approach analyzes the texture and gradient structures of the facial images using a set of low-level feature descriptors, fast linear classification scheme and score level fusion. Compared to many previous works, the proposed approach is robust and does not require user-cooperation. In addition, the texture features that are used for spoofing detection can also be used for face recognition. This provides a unique feature space for coupling spoofing detection and face recognition.

This work proposes a new face anti- spoofing method based on color texture analysis. I analyze the joint color-texture information from the luminance and the chrominance channels is analyzed using a color local binary pattern descriptor. More specifically, the feature histograms are extracted from each image band separately.

The majority of research on non-intrusive software-based face spoofing detection schemes has focused on the analysis of luminance information in face images, avoiding the Chroma component, which can be very useful for distinguishing fake faces from genuine ones. This paper explains a novel and appealing method to detect face spoofing across color texture analysis. By extracting complementary low-level feature descriptions from different color spaces, we exploit the joint color texture information from the luminance and chrominance channels. The extracted Histogram and color texture information will be classified using SVM classifier and CNN.

Keywords— Color texture analysis, Deep Learning, Face recognition, Machine Learning, spoofing Detection, presentation attack.

CONTENTS

TABLE OF CONTENTS

LIST OF FIGURE

INTRODUCTION

CHAPTER - 1

INTRODUCTION

1.1 ABOUT THE PROJECT

Like all biometrics solutions, facial recognition technology measures and matches the unique characteristics for the purposes of identification or authentication. Often leveraging adigital or connected camera, facial recognition software can detect faces in images, quantify their features, and then match them against stored templates in a database. Face-scanning biometrictech is incredibly versatile and this is reflected in its wide range of potential applications.

Face biometrics have the potential to be integrated anywhere you can find a modern camera. A growing number of law enforcement agencies around the world are using biometric software to scan faces in CCTV footage, as well as to identify persons of interest in the field. Border authorities, meanwhile, are increasingly using facial recognition to verify the identities of travelers, especially at airports. And in the mobile sector, face-scanning systems have become a hugely popular means of unlocking smart phones – as can be seen in Apple's pioneering Face ID system on its newer phones – as well as verifying payments and signing into mobile apps.

While controversy has emerged over how law enforcement authorities use facial recognition, many police officials have argued that the technology helps them to fight crime, andit can also be used to identify missing persons and the trafficking. Meanwhile, as a contactless biometric solution that's easy to deploy in consumer devices, face recognition is showing the public just how convenient strong authentication can be. Facial recognition doesn't just deal with hard identities, but also has the ability to gather demographic data on crowds. This has made face biometrics solutions increasingly sought after in the retail marketing industry.

Now a Day, it is known that most of existing face recognition systems is vulnerable to spoofing attacks. A spoofing attack occurs when someone tries to bypass a face biometric system by presenting a fake face in front of the camera. For instance, in, researchers inspected the threat of the online social networks based facial disclosure against the latest version of six commercial face authentication systems (Face Unlock, Face lock Pro, Visidon, Veriface, Luxand Blink and

Fast Access). While on average only 39% of the images published on social networks can be successfully used for spoofing, the relatively small number of usable images was enough to fool face authentication software of 77% of the 74 users.

Also, in a live demonstration during the International Conference on Biometric (ICB 2013), a female intruder with a specific make-up succeeded in fooling a face recognition system. These two examples among many others highlight the vulnerability of face recognition systems to spoofing attacks. Assuming that there are inherent disparities between genuine faces and artificial material that can be observed in single images (or a sequence of images), many anti-spoofing techniques analyzing static (and dynamic) facial appearance properties have been proposed.

The key idea is that an image of a fake face passes through two different camera systems and a printing system or a display device, thus it can be referred to in fact as a recaptured image. As a consequence, the observed fake face image is likely to have lower image quality compared toa genuine one captured in the same conditions due to e.g. lack of high frequency information.

Furthermore, the recaptured images may suffer from other quality issues, such as content-independent printing artifacts or video noise signatures. In the literature, the facial appearance analysis based methods are usually referred to as texture or image quality analysis based techniques because the aforementioned properties can be considered as variations in the facial texture information or image quality.

The recapturing process described above introduces also inherent disparities in the color information between a genuine face and a recaptured face image. This is due to the used spoofing medium (printed photograph, display device or mask) dependent gamut and other imperfections in the color reproduction, e.g. printing defects or noise signatures.

It is no surprise that cybercrime is on the rise in our increasing digital world. Many companies are now exploring biometric face recognition as viable security solution. The general public has immense need for security measures against spoof attack. Biometrics is the fastest growing segment of such security industry. Some of the familiar techniques for identification are facial recognition, fingerprint recognition, handwriting verification, hand geometry, retinal andiris scanner. Among these techniques, the one which has developed rapidly in recent years is face recognition technology and it is more direct, user friendly and convenient compared to other methods. This innovative technology shows a lot of promise and change the way in which we can access sensitive information. But as promising as facial recognition is it does have flaws. User photos can easily be found on social networking sites and images can be spoofed. This iswhere

The need of anti-spoofing comes into play. Face anti spoofing is the task of preventing false facial verification by using a photo, video/substitute for an authorized person's face.

1.2 WHY FACE BIOMETRIC IS IMPORTANT

Facial recognition is in the limelight today more than ever. Previous historical events have resulted in a quick hike in face recognition investments.

Given the worldwide COVID-19 epidemic, we may anticipate more investment in biometric technologies like facial recognition. Given the extremely infectious nature of COVID-19, there is a strong emphasis on contactless interactions.

The principal use of face recognition technology continues to be security solutions. Facial recognition is recognized as one of the most accurate and easy ways for establishing individual identity across a wide range of sectors.

1.3 ARCHITECTURE

Biometric sensors or access control systems are classified into two types such as Physiological Biometrics and Behavioral Biometrics. The physiological biometrics mainly include face recognition, fingerprint, hand geometry, Iris recognition, and DNA. Whereas behavioral biometrics include keystroke, signature and voice recognition.

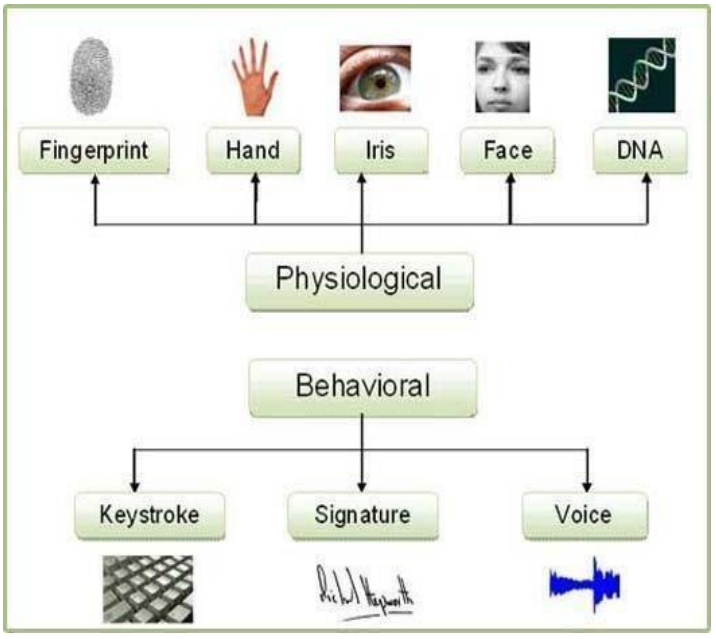


Figure 1.3.1: Biometric system Architecture

Fingerprint Recognition

Fingerprint Recognition includes taking a fingerprint image of a person and records its features like arches, whorls, and loops along with the outlnes of edges, minutiae, and furrows. Matching of the Fingerprint can be attained in three ways, such as minutiae, correlation, and ridge

- Minutiae based fingerprint matching stores a plane includes a set of points and the set of points are corresponding in the template and the I/p minutiae.
- Correlation-based fingerprint matching overlays two fingerprint images and the association between equivalent pixels is calculated.
- Ridge feature-based fingerprint matching is an innovative method that captures ridges, as minutiae-based fingerprint capturing of the fingerprint images is difficult in low quality.

o capture the fingerprints, present methods employ optical sensors that use a CMOS image sensor or CCD; solid-state sensors work on the principle of transducer technology using thermal, capacitive, piezoelectric sensors or electric field ; or ultrasound sensors work on echography in which the sensor sends acoustic signals through the transmitter near the finger and captures the signals in the receiver. The scanning of the fingerprint is very stable and also reliable. It safeguards entry devices for building door locks and access of computer network is becoming more mutual. At present, a small number of banks have initiated using fingerprint readers for approval at ATMs.

Face Recognition

A face recognition system is one type of biometric computer application that can identify or verify a person from a digital image by comparing and analyzing patterns. These biometric systems are used in security systems. Present facial recognition systems work with face prints and these systems can recognize 80 nodal points on a human face. Nodal points are nothing but endpoints used to measure variables on a person's face, which includes the length and width of the nose, cheekbone shape, and eye socket depth.

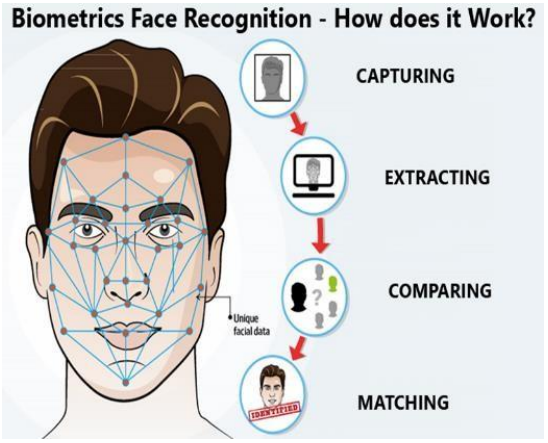


Figure 1.3.2: Face Biometric system

- Facial recognition is a way of identifying or confirming an individual's identity using their face.
- Facial recognition systems can be used to identify people in photos, videos, or in real-time.

Face recognition systems work by capturing data for the nodal points on a digital image of a person's face and resulting data can be stored as a face print. When the conditions are favorable, these systems use face prints to identify accurately. Currently, these systems focus on smartphone applications which include personal marketing, social networking, and image tagging purposes. Social sites like FB uses software for face recognition to tag the users in photographs. This software also increases marketing personalization. For instance, billboards have been designed with integrated software that recognizes the ethnicity, gender and estimated age of onlookers to deliver targeted marketing.

Iris Recognition

Iris recognition is one type of bio-metric method used to identify the people based on single patterns in the region of ring-shaped surrounded the pupil of the eye. Generally, the iris has a blue, brown, gray or green color with difficult patterns that are noticeable upon close inspection. Please follow the below link to know more about iris recognition technology. Please follow the link to know more about IRIS Recognition Technology.

Figure 1.3.3: Iris Recognition

Voice Recognition

Voice recognition technology is used to produce speech patterns by combining behavioral and physiological factors that can be captured by processing speech technology. The most important properties used for speech authentication are nasal tone, fundamental frequency, inflection, cadence. Voice recognition can be separated into different categories based on the kind of authentication domain, such as a fixed text method, in the text-dependent method, the text-independent method, and conversational technique. Please follow the link to know more about Voice Recognition technology.

Figure 1.3.4: Voice Recognition

Signature Recognition

Signature recognition is one type of biometric method used to analyze and measure the physical activity of signing like the pressure applied, stroke order and speed. Some biometrics are used to compare visual images of signatures. Signature recognition can be operated in two different ways, such as static and dynamic.

Figure 1.3.5: Signature Recognition

In static mode, consumers write their signature on paper, digitize it through a camera or an optical scanner. This system identifies the signature examining its shape.

In dynamic mode, consumers write their signature in a tablet which is digitized, which obtains the signature in real-time. Another option is gaining by means of stylus-operated PDAs. Some biometrics also operates with smart-phones with a capacitive screen, where consumers can sign using a pen or a finger. This type of recognition is also known as "on-line".

Thus, this is all about biometric sensors which can be used by several organizations to increase the level of security and also to protect their data and copyrights. We hope that you have got a better understanding of this concept. Furthermore, any doubts regarding the concept or electrical and electronics projects. Please comment in the comment section below. Here is a question for you, what are the applications of biometric sensors?

Photo Credits:

- Biometric Sensor data-structure
- Types of Biometric Sensorslidesharecdn
- Fingerprint Recognition wordpress
- Face Recognition thirdeyesystems
- Iris Recognition bioenabletech
- Voice Recognition termcoord
- Signature Recognition biometricsintegrated

- Why I am taking Face is, it will preventing crimes and increasing safety and security to reducing unnecessary human interaction and labor.
- In some instances, it can even help support medical efforts.

1.4 APPLICATION USED

Many traders believe that biometrics are appropriate for government use only, but they are rapidly learning that the purposes of biometrics expand far ahead of the government employ completely. In this, we will mention the top 5 applications of biometric authentication across the sphere — places where the biometric technology is used to make more security and ease for daily citizens.

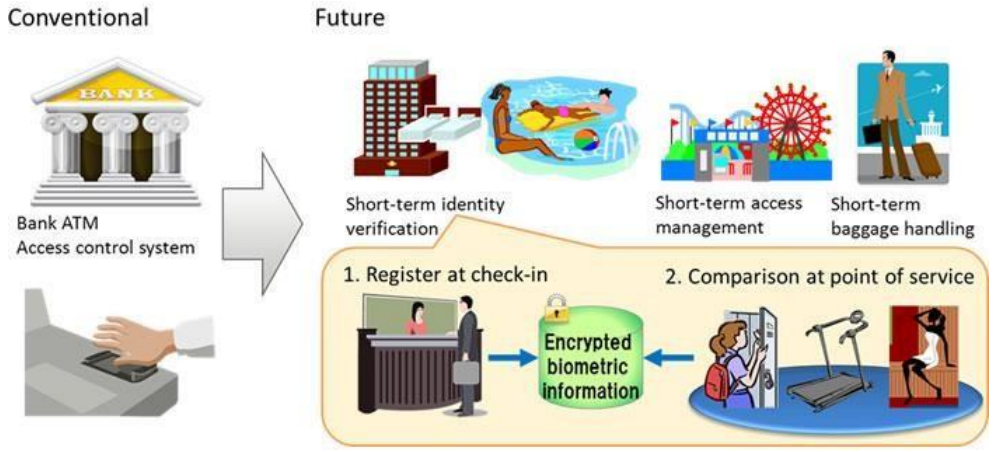


Figure 1.4.1: Applications of Biometric Authentication

- Security in Airport
- Time and Attendance
- Enforcement of Law
- SSO (Single Sign-On) & Access Control
- Transaction verification in Banking

Biometric identification systems offer higher electronic security, convenience, accountability, and accurate audit trails – all attributes that motivate businesses to research and implement the technology for their use. We believe that as time moves forward, we will see the implementation of biometric technology continue to grow and be used in even more areas that touch our lives.

1.5 CHALLENGES IN FACE BIOMETRICS SYSTEM

To achieve a desired level of reliability and accuracy, face recognition should be performed based on intrinsic properties of the face only, mainly, 3D shape and reflectance of the facial surface. Extrinsic properties, including hairstyle, expression, posture, and environmental lighting, should be minimized. Environmental lighting may or may not be controllable depending on the operation. The ideal case would be that both input and enrollment face images are subject to the same lighting conditions, including, lighting direction and lighting intensity. The problem of uncontrolled environmental

Lighting is the first challenge to overcome for face based biometric applications.

Most commercial face recognition systems have a common problem that the accuracy drops when lighting conditions of input and enrollment are different. Regarding the user factors, most existing systems allow a limited degree of variations in expression, facial hair, and facial-ware. Also these properties can be controlled in cooperative user applications. For example, the user can be advised not to play exaggerated expressions, or not to wear sun-glasses. However, facial aging leads to changes in intrinsic properties (shape and skin reflectance) that progress as people age. Face recognition under significant aging of facial appearance is an unsolved problem.

1.6 SPOOFING ATTACK

Spoofing attack is a situation in which a person or program successfully identifies as another by falsifying data, to gain an illegitimate advantage.

How does spoofing work?

Spoofing is a cyber-attack that occurs **when a scammer is disguised as a trusted source to gain access to important data or information**.

Spoofing can happen through websites, emails, phone calls, texts, IP addresses and servers.

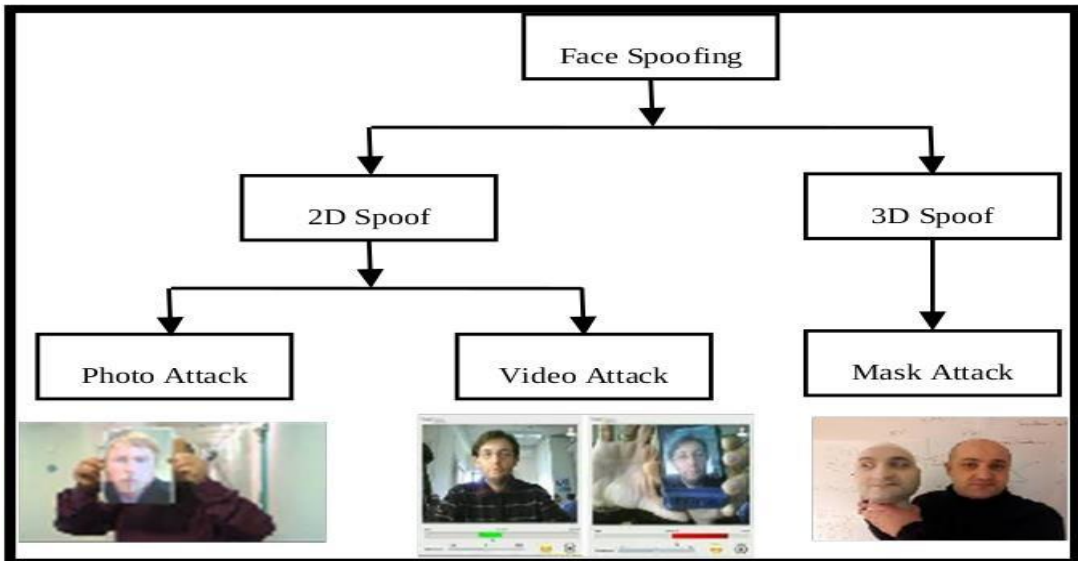


Figure 1.6.1: Types of Spoofing

Photo attacks: a photo attack consists of displaying a photograph of the attacked identity to the sensor of the face recognition system.

Video attacks: an attacker could play a video of the legitimate user in any device that reproduces video and then presents it to the sensor/camera.

3D Mask Attacks: in this type of attack, the attacker builds a 3D reconstruction of the face and

Presents it to the sensor/camera.

Other attacks: makeup, surgery

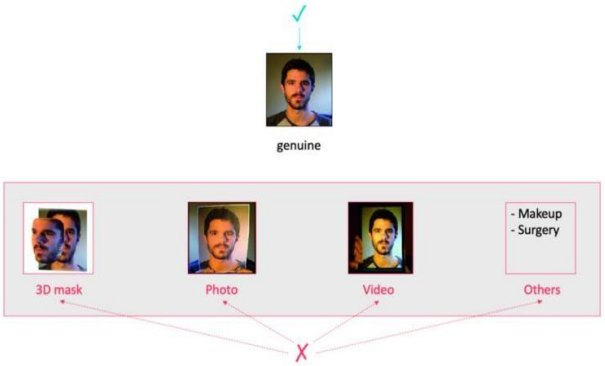


Figure 1.6.2: Verification Architecture

1.7 MOTIVATION AND JUSTIFICATION

According to National Institute of Standards and Technology, US department of commerce, "Recommended the Presentation Attack Detection (PAD) and Mitigation of threats due to spoof attacks for new biometric test"- Apr 5, 2022.

- Even though the face recognition biometrics has n number of advantages, it has same amount of disadvantages too
- Thus, there is a huge need for developing a better face anti-spoofing biometric system.

1.8 PROBLEM STATEMENT

To detect face spoofing attack in face biometric authentication system by analyzing texture and color quality of face images.

1.9 OBJECTIVES

- To analyze the luminance of the face images,
- To discard the chrominance information which can be useful for discriminating fake faces from genuine ones.

1.9.1 Primary Objective

- To develop a model for face spoofing attacks in biometric authentication system.

1.9.2 Secondary Objective

- To enhance spoofing attack detection rate by performing gamma correction technique.
- To make use of LBP, HOG for feature extraction.
- To perform Comparative analysis of SVM and CNN in face spoofing attack detection.

SYSTEM CONFIGURATION

CHAPTER - 2
SYSTEM CONFIGURATIONS

2.1 HARDWARE SPECIFICATION

PROCESSOR	: Intel i7 and above
RAM	: 8 GB
HARD DISK CAPACITY	: 1TB

2.2 SOFTWARE SPECIFICATION

OPERATING SYSTEM	: Windows10
PACKAGE	: Anaconda
Front End	: Python (3.8)

2.3 ABOUT THE SOFTWARE

Anaconda is a Python and R computer language distribution aimed at making management system to ensure and deployment easier in data science (science, deep learning software, large-scale data analysis, predictive analytics, and so on). For Pc, Linux, and macOS, the distribution offers data-science packages. Anaconda, Inc., created by Peter Wang and Travis Oliphant in 2012, is in charge of its development and maintenance. Anaconda Distributed or Anaconda Individual Edition is two Anaconda, Inc. products, whereas Anaconda Team Version and Anaconda Edition, both are not free, are two more Anaconda, Inc. products.

Conda, Anaconda's package management system, keeps track of package versions. This packages manager is spun out like a distinct outdoor package because it proved to be helpful in and outside of Python. Miniconda is a compact, bootstrapped version of Anaconda that only

Includes conda, Python, and the programmes they depend on, as well as a few more packages.

2.3.1 ANACONDA

Anaconda is a Python and R programming language distribution aimed for simplifying package management and deployment in programming (data science, machine learning applications, large-scale data processing, predictive analytics, and so on). Data-science packages for Windows, Linux, and macOS are included in the release. Anaconda, Inc., created by Peter Wang and Travis Oliphant in 2012, is responsible for its development and maintenance. Anaconda Distribution or Anaconda Individual Edition is other Anaconda, Inc. goods, whereas Anaconda Team Edition and Anaconda Enterprise Edition, both are not free, are other Anaconda, Inc. products.

It is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage anconda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, mac OS and Linux.

The following applications are available in Anaconda Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

2.3.2 JUPYTER NOTEBOOK

Jupyter Notebook is a popular programming environment an open source, interactive web tool that lets user create and share documents with interactive calculations, code, and graphics, among other things. Users can create interactive "story" that can edit and share by combining data, code, and visuals into a single notebook. Notebooks are documents that include both computer code (such as Python) and other text elements including paragraphs, markdown,figures, and links, among other things. Jupyter is a programming notebook is well-known and well-documented, with a simple interface for generating, editing, and executing notebooks.

The notebook is accessed through a web application known as the "Dashboard" or"Control panel", which displays local files and allows users to view notebook pages and run codesnippets. The results are well formatted and presented in the browser. The kernel is the notebook's other component. The kernel is a types of "computational engine" that runs the code in the notebook. It's similar to the application's backend. Jupyter notebook, the IPython kernel (jupyter was originally known as IPython notebook) is utilized to run Python code.

2.3.3 PYTHON PACKAGE

Common Importing Python package using Jupyter Notebook are:

The use of Jupyter Notebooks to import code is a common problem.

a) import pandas as pd

Pandas is typically installed using the pd alias. alias: In Python, an alias is a different name for the same item. Instead of pandas, the Pandas package is now known as pd.

b) import numpy as np

Python is told to introduce the NumPy library to the current environment via the import numpy section of the code. The as np subsection then instructs Python to assign the alias np to NumPy. Instead of typing numpy, it can use NumPy functions besides typing np. function name.

c) import matplotlib.pyplot as plt

Matplotlib is a Python package that allows it to create static, animated, and interactive visualisations. Matplotlib makes simple things simple and difficult things possible that can be Create plots that are suitable for publication, Customize the visual style and layout of dynamic figures which can zoom, pan, and update.

d) from sklearn.metrics

Metrics for classification to measure classification performance, the sklearn. metrics module implements several loss, score, and functionalities. Some metrics may demand positive class probability estimations, confidence variables, or binary decision values.

e) from sklearn.model_selection import train_test_split

Train test split is a Sklearn model selection method that divides information arrays into two subsets: test and train information. With this function, there should be no need to manually divide the dataset. By default, the Sklearn train test split generates random differences for the two subsets.

f) from sklearn.metrics import preprocessing

sklearn. preprocessing package contains several common transfer functions and transformer classes for converting the raw relevant features into an interpretation that is more suitable for downstream estimators. In general, the standardization of the set of data benefits supervised learning.

g) from sklearn.preprocessing import MinMaxScaler

sklearn.preprocessing.MinMaxScaler class sklearn.preprocessing. MinMaxScaler (feature range=0, 1) (copy=True) (feature [source] Scales each attribute to a set range to transform it. This estimate scales and translations each feature separately so that it falls inside the training set's provided range, i.e. among zero and one.

h) from sklearn.preprocessing import standardscaler

Instead of removing the mean and scaling to unit variance, StandardScaler standardises afeature. Divide all of the numbers by the standard deviation to get unit variance. The formal concept of scale that I introduced earlier does not apply to StandardScaler.

i) from sklearn import metrics

To measure classification performance, the sklearn. metrics module implements several loss, score, and utility functions. Some metrics may demand positive class probability estimations, confidence values, or binary decision values.

j) from sklearn.metrics import classification_report

In machine learning, a classification report is a performance evaluation indicator. It's used to demonstrate trained classification model's precision, recall, F1 Score, and support.

k) from sklearn.metrics import accuracy_score

This method computes subset accuracy in multilabel classification: any subset of label forecast for a sample must match exactly the corresponding set of keywords in y true. More information can be found in the User Manual. Labels that is true to the ground (accurate).

I) from sklearn.metrics import precision_score

sklearn.metrics.precision score. Precision is defined as tp / (tp + fp), in which tp is the number of actual true positives and fp seems to be the proportion of false positives. Precision is intuitively defined as the classifier's ability to avoid labeling something negative sample as positive.

M) from sklearn.metrics import recall_score

sklearn.metrics.recall score is used to calculate recall is defined as the proportion tp / (tp + fn), where tp represents the number of true positive rate and fn represents the number of false negatives rate.

The recall is implicitly the classifier's ability to find all positive samples. The best possible value 1 but worst possible value is 0.

n) from sklearn.metrics import f1_score

Using the f1 score framework, we must first import the package listed below. F1 score is a component of the sklearn.metrics package. Here is the full syntax for the F1 score function. The required parameters are y true and y pred. Others are optional parameters that are not required.

REVIEW OF LITERATURE

CHAPTER - 3

REVIEW OF LITERATURE

S.No	Title of the Paper	Methods Used	Dataset	Results
1.	Face anti-spoofing based on color texture analysis 2018	• Local Binary Patterns (LBP) • Support Vector Machine(SVM)	(CASIA-FA) Replay Attack	81.4 %
2.	Face Anti-Spoofing Using Texture-Based Techniques and Filtering Methods 2019	• Difference of Gaussian (DOG) filtering method • Support vector machine (SVM) • LTP (Local Ternary Pattern) features	NUAA photo-imposter database	LTP-91.1% DoG- LBPV 99.22 %
3.	Texture and quality analysis for face spoofing detection 29 January 2021	• Novel method - joint exploitation of entropy-based texture and quality features.	Replay attack	• 98.2%
4.	Texture based Face recognition using GLCM and LBP schemes 2020	• (GLCM) LocalBinary Patterns named as GOL texture feature technique for face	(ORL-faces) Georgia Tech (GT-faces)	• 98%

		classification		
5.	LGLG-WPCA: An Effective Texture-based Method for Face Recognition 7 Jun 2019	• Gabor wavelet • Multivariate Gaussian • Log-Euclidean Gaussian • Face frontalization • Occlusion recovery • Illumination compensation or elimination	Deep Convolutional Network (DCNN)-own dataset	• 97.44%
6.	Texture and quality analysis for face spoofing detection Sep 2021	• Novel method - joint exploitation of entropy-based texture.	Replay attack database	• 95.2%
7.	Face Anti-spoofing Method Using Color Texture Segmentation on FPGA May 2021	• LBP, • The Cb, S, and H bands are used from the color spaces.	CASIA-FASD dataset	• 98.03%

LITERATURE EXPLANATION:

[1] Face spoofing identification primarily aimed to evaluate the illumination of face images, eliding luminance informationwhich is useful in distinguishing genuinefrom fake faces. In this study, we suggest a new faceanti-spoofingtechnique based on colourtexture analysis.

[2] This project presents an interesting face create a fake detection approach based on an examinationof comparison as well asinteractive change from time to time in both captured and spoofed photos. Correct picture spoofing identification isexpected in this case.

[3] This project proposed a novel method for exploring the robustness of face spoofing detectorby ability to combine entropy-based texture and quality features. Theeffectiveness of this method is assessed using a publicly available datasetidentified as the "replay attack database."

[4] The proposed methodlearns face features froman embedded multivariate Gaussian in the Gabor wavelet domain; it needsto perform much under adverse conditions such asmodifying poses, skin aging, and uneven illumination.

[5] This article addresses a novel as well as appealing colour texture analysistechnique for detectingface spoofing. To extract the combined chromatexture features from thechrominance and chrominance channels, weuse matching low in fat and high explanations from different colour spaces.

[6] I Face Convolutional Neural Anti-spoofing -As a classification of two-class issues, this paper is opposed to face spoofing. We propose a technique for trying to identify face spoofing utilising color -shape data from faceimages in this paper, which is also based on convolutional neural network (cnn. To analyse the color-texture data in conjunction with chrominance and colourdifference channels, a local binary descriptor isused. The CASIA-FASD dataset was used to validate the proposed scheme. The proposed method outperformedprevious studies' state-of- the-art methods.

METHODOLOGY

CHAPTER - 4

METHODOLOGY

The proposed framework has four phases. Phase I demonstrates the dataset used. In Phase II, the pre-processing techniques is reducing the over brightness of an image using gamma correction are explained. In phase III, the Feature Extraction for detecting the faces using HOG and LBP. In phase IV the Machine learning and Deep learning detection is implemented to detect spoofed person.

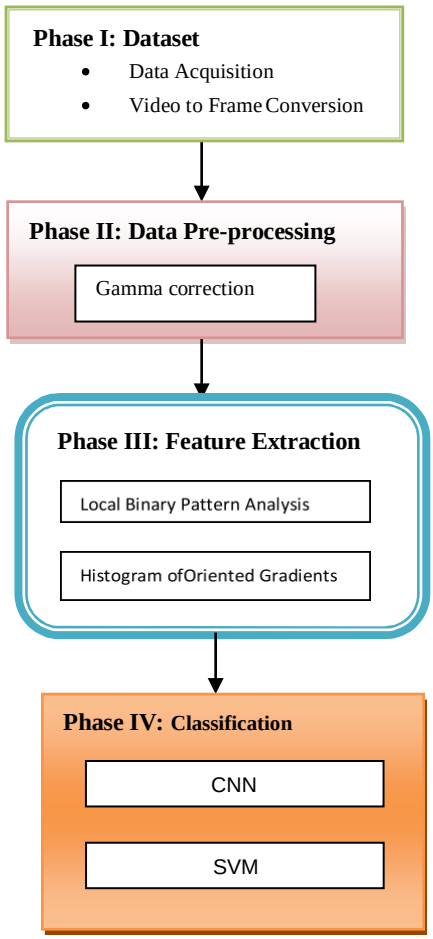


Figure 4. 1: shows the proposed methodology using supervised learning-based spoof detection.

Phase I: Dataset

To perform the above-mentioned framework, “Reply Mobile Attack” dataset (Texture and Quality Analysis for Face spoofing Detection-2021)isused.Replay-Mobile is a dataset for face recognition and presentation attack detection (anti-spoofing).The dataset consists of 1190 video clips of photo and video presentation attacks (spoofing attacks) to 40 clients, under different lighting conditions. The front-camera produces color videos with a resolution of 720 pixels(width) by 1280 pixels (height) and saved in ".mov" file-format.Real-accesses have been performed by the genuine user (presenting one's true face to the device). Attack-accesses have been performed by displaying a photo or a video recording of the attacked client, for at least 10 seconds.

Real client accesses have been recorded under five different lighting conditions (controlled, adverse, direct, lateral and diffuse). In addition, to produce the attacks, high- resolution photos and videos from each client were taken under conditions similar to those in their authentication sessions (lighton,lightoff).

The 1190 real-accesses and attacks videos were then grouped in the following way:

- **Training set**: contains 120 real-accesses and 192 attacks under different lighting conditions;
- **Development set**: contains 160 real-accesses and 256 attacks under different lighting conditions;
- **Test set**: contains 110 real-accesses and 192 attacks under different lighting conditions;
- **Enrollment set**: contains 160 real-accesses under different lighting conditions, to beused **exclusively** for studying the baseline performance of face recognition systems. (This set is again partitioned into 'Training', 'Development' and 'Test' sets.)

Phase II: Data pre-processing

In the selected dataset, the input images are taken from video recordings. Face images are preprocessed to normalize the illumination. In this gamma correction is used. Different camera or video recorder devices do not correctly capture luminance. Different display devices such as monitor, phone screen, TV do not display luminance correctly neither. So, one need to correct them, therefore the gamma correction functions. Gamma correction function is used to correct image's luminance.

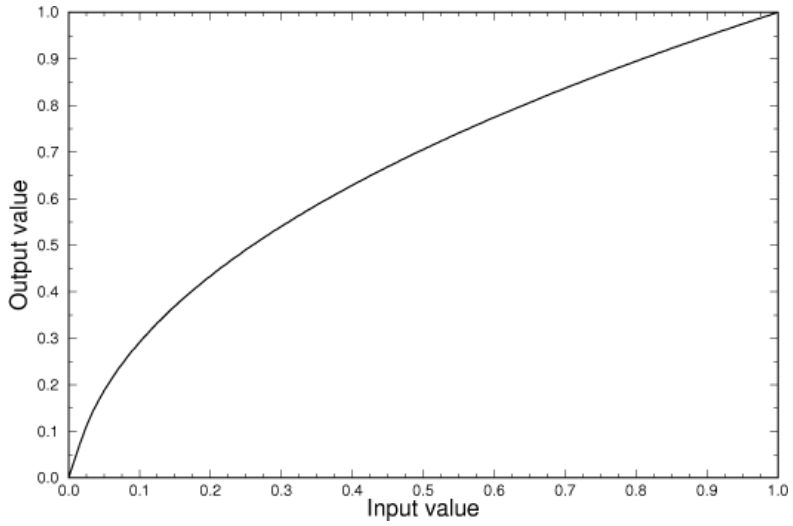


Figure 4.2: Color balance of a gamma corrected image

If an image is under or over gamma corrected, this also affects the color balance. Over correction (in addition to making mid-tones too light) shifts colors towards neutral grey, while under correction (in addition to making mid-tones too dark) shifts colors towards the display primaries.

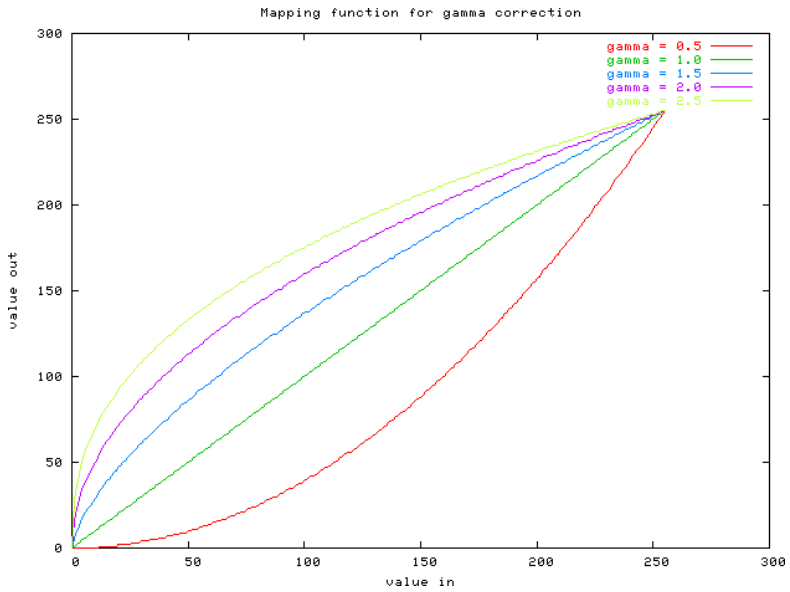


Figure 4.3: Mapping function for gamma correction

The whole thing about the gamma complexity is of 2 main causes:

- Human perception of the intensities of color or brightness does not have a linear relationship, with respect to the values of intensity of light waves in physics. That is, we are more sensitive to some color or brightness range.
- The brightness of image display devices, such as CRT, does not have a linear relation of its voltage input.

Output luminance = gamma Correction Function [Input luminance]

Gamma Correction is one of the suitable techniques to handle the brightness and control the quality of an image data. Figure 2: illustrates the training set instance before and after applying gamma correction technique.

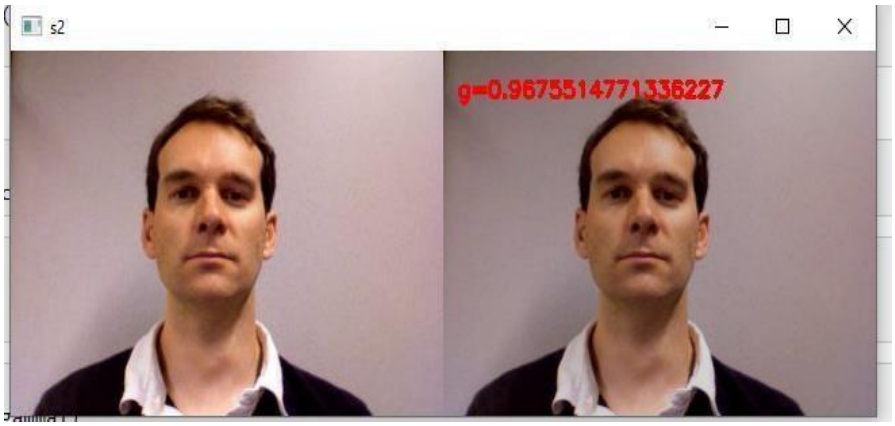


Figure 4.4: Before and after applying the Gamma Correction.

Phase III: Feature Extraction

The LBP was introduced by Ojala et al; it has been demonstrated as a powerful grayscale invariant texture descriptor in the literature. An LBP operator integrates characteristics of structural and statistical texture analysis. The LBP illustrates texture using microprimitives and their statistical placement rules. The LBP performs on a pixel basis and illustrates the eight surrounding pixels in binary code. The LBP subsequently summarizes all codes into a histogram, which facilitates the extraction of a texture feature. Therefore, a 256- texture pattern for a 3 × 3 neighboring would be produced. Consider the following matrix:

The possible 256-bit pattern that is produced by (3) will be used to construct a histogram, which will facilitate the process of texture description. Texture is a significant characteristic of different types of images and is involved in several images from multispectral images to microscopic images. The significance of texture is depicted in numerous computer vision and image analysis applications. Recently, different discriminative local texture descriptors have been proposed; the most common descriptor is local binary pattern. The local binary pattern (LBP) was originally presented as a texture descriptor [5]. The LBP has been utilized in several domains of computer vision, such as face recognition and facial expression recognition, to model motion and actions. Multiple modifications have been conducted on the original LBP to fit other tasks. The LBP has contributed toward significant progress in the area of texture analysis, where various applications that range from 2D texture to 3D texture have been examined. The LBP can be

viewed as a unifying method for the classical divergent statistical and structural models of texture analysis. The key success factor behind the LBP is its accurate monotonic grayscale changes, such as illumination variations. In addition, the LBP has other advantage-simple computations-which render it competitive in the area of real-time image analysis. The LBP depends on the assumption that emphasizes that texture has two complementary perspectives: pattern and strength. The pixels of an image are being annotated by a specific threshold that compares neighboring pixels with the center. The result will be represented as a binary number. This resulting value will be used as a texture descriptor.

The first table contains a 3×3 pixel, which represents an image portion. The LBP focuses on the center, in which all surrounding pixels will be compared against the center. This comparison attempts to identify the smaller and greater values. Thus, the pixels that have greater values than the center will be encoded as 1, and the pixels that have values smaller than the center will be encoded as 0, as shown in the second table (i.e., threshold). The pattern can be extracted as '10001111' and will be utilized as a texture feature for learning purposes. The third table provides an assumption by assigning weights to all pixels (which are powers of 2).

Local Binary Pattern Analysis-The LBP descriptor is a highly discriminative gray-scale texture descriptor. For each pixel in an image, a binary code is computed by thresholding a circularly symmetric neighborhood with the value of the central pixel. Finally, a histogram is created to collect the occurrences of different binary patterns. LBP was originally intended to handle gray- scale images but was later extended to exploit also color information. In a simple yet efficient color LBP descriptor was proposed. The LBP operator is applied on each color band. The LBP pattern of a pixel (x, y) extracted from the image band (i) can be written as follows:

$$\mathrm{LBP}(x_c, y_c) = \sum_{i=0}^{7} s(g_i - g_c)2^i$$

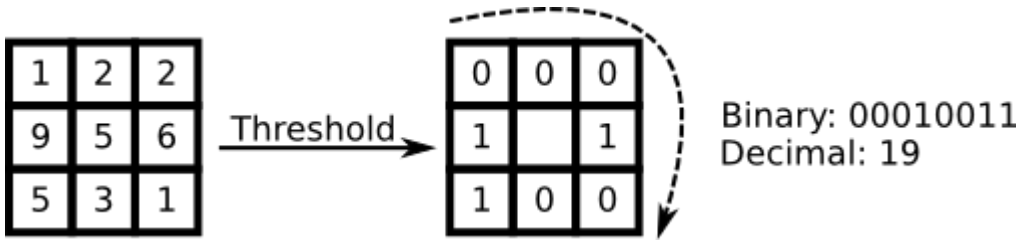
$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

gc- the intensity value of the central pixel

gi- the intensity of the neighboring pixel with index i

To detect spoofing attacks, the color LBP features extracted from face images are fed into a Support Vector Machine (SVM) classifier.

Local Binary Pattern (LBP) is a very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. The LBP feature vector, in its simplest form, is created in the following manner:



- Divide the examined window into cells (e.g. 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counterclockwise.
- In the above step, the neighbors considered can be changed by varying the radius of the circle around the pixel, R and the quantization of the angular space P.
- Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
- Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.
- Optionally normalize the histogram.

- Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

The feature vector can now then be processed using some machine-learning algorithm to classify images. Such classifiers are often used for face recognition or texture analysis.

Histogram of Oriented Gradients

Histogram of Oriented Gradients, also known as HOG, is a feature descriptor like the Canny Edge Detector; SIFT (Scale Invariant and Feature Transform). It is used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in the localized portion of an image. This method is quite similar to Edge Orientation Histograms and Scale Invariant a Feature Transformation (SIFT). The HOG descriptor focuses on the structure or the shape of an object. It is better than any edge descriptor as it uses magnitude as well as angle of the gradient to compute the features. For the regions of the image it generates histograms using the magnitude and orientations of the gradient.

Calculate HOG Descriptor vector

- To calculate the final feature vector for the entire image patch, the 36×1 vectors are concatenated into one giant vector.
- So, say if there was an input picture of size 64×64 then the 16×16 block has 7 positions horizontally and 7 position vertically.
- In one 16×16 block we have 4 histograms which after normalization concatinate to form a 36×1 vector.
- This block moves 7 positions horizontally and vertically totalling it to 7×7 = 49 positions.
- So when we concatenate them all into one gaint vector we obtain a 36×49 = 1764 dimensional vector.
- This vector is now used to train classifiers such as SVM and then do object detection.

Phase IV: Classification

CNN based Face Anti-spoofing

In the last few years of the IT industry, there has been a huge demand for once particular skill set known as Deep Learning. Deep learning a subset of Machine Learning which consists of algorithms that are inspired by the functioning of the human brain or the neural networks.

These structures are called as Neural Networks. It teaches the computer to do what naturally comes to humans. Deep learning, there are several types of models such as the Artificial Neural Networks (ANN), Auto encoders, Recurrent Neural Networks (RNN) andReinforcement Learning. But there has been one particular model that has contributed a lot in thefield of computer vision and image analysis which is the Convolutional Neural Networks (CNN) or the ConvNets.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

The term 'Convolution" in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

Basic Architecture

There are two main parts to CNN architecture

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

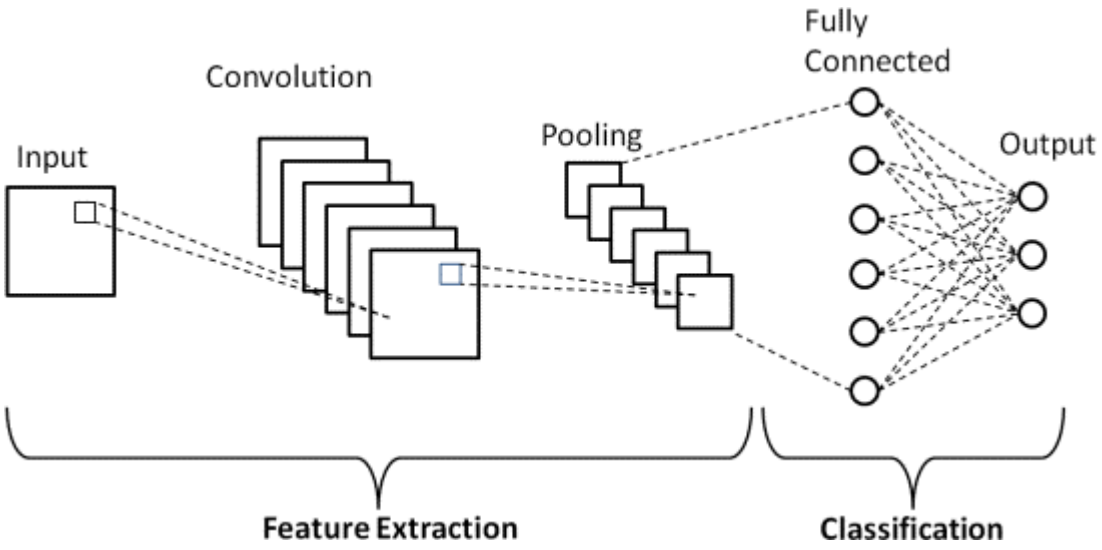


Figure 4.5: CNN Architecture

Convolution Layers

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

1. **Convolutional Layer**

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

The first layer in a CNN network is the *CONVOLUTIONAL LAYER*, which is the core building block and does most of the computational heavy lifting. Data or imaged is convolved

using filters or kernels. Filters are small units that we apply across the data through a sliding window. The depth of the image is the same as the input, for a color image that RGB value of depth is 4; a filter of depth 4 would also be applied to it. This process involves taking the element-wise product of filters in the image and then summing those specific values for every sliding action. The output of a convolution that has a 3d filter with color would be a 2d matrix.

2. Pooling Layer

The ***POOLING LAYER***, It is applied through every layer in the 3d volume.

Typically there are hyper parameters within this layer:

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer

1. **The dimension of spatial extent:** which is the value of n which we can take N cross and feature representation and map to a single value
2. **Stride:** which is how many features the sliding window skips along the width and height

A common **POOLING LAYER** uses a 2 cross 2 max filter with a stride of 2, this is a non-overlapping filter. A max filter would return the max value in the features within the region. Example of max pooling would be when there are 26 across 26 across 32 volumes, now by using a max pool layer that has 2 cross 2 filters and a stride of 2, the volume would then be reduced to 13 crosses, 13 crosses 32 feature map.

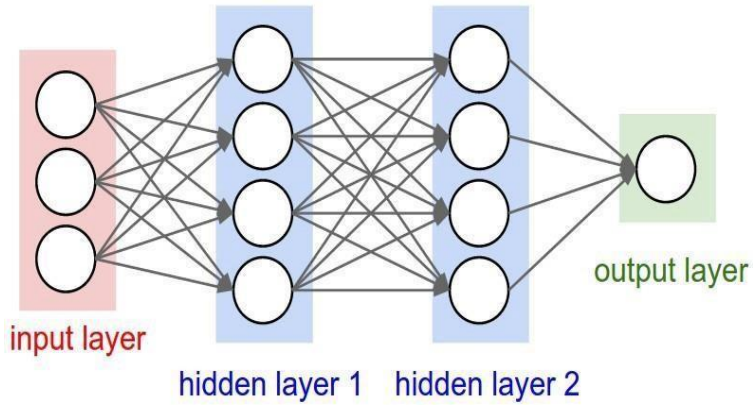


Figure 4.6: CNN Architecture layer

3. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

Lastly, is the ***FULLY CONNECTED LAYER,*** This involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing. With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the output.

4. Dropout

Usually, when all the features are connected to the FC layer, it can cause over fitting in the training dataset. Over fitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.

To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

5. Activation Functions

- Finally, one of the most important parameters of the CNN model is the activationfunction. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should notat the end of the network.
- It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions has a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred and for a multi-class classification, generally softmax us used.
- Now, the best way to explain a convolutional layer is to imagine a flashlight that is shining over the top left of the image. In order to understand how this works, imagine as if a flashlight shines its light and covers a 5 x 5 area. And now, let's imagine this flashlight sliding across all the areas of the input image.
- This flashlight is called a filter (or sometimes referred to as a neuron or a kernel) and the region that it is shining over is called the receptive field. This filter is also an array of numbers (the numbers are called weights or parameters).

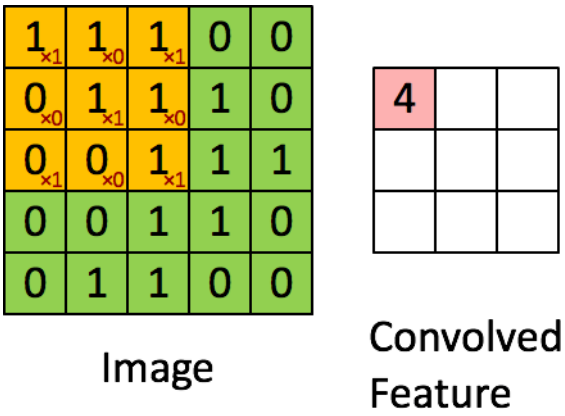


Figure 4.7: Feature Conversion

6. LeNet-5 CNN Architecture

- In 1998, the LeNet-5 architecture was introduced in a research paper titled "Gradient-Based Learning Applied to Document Recognition" by Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. It is one of the earliest and most basic CNNarchitecture.
- It consists of 7 layers. The first layer consists of an input image with dimensions of 32×32. It is convolved with 6 filters of size 5×5 resulting in dimension of 28x28x6. The second layer is a Pooling operation which filters size 2×2 and stride of 2. Hence the resulting image dimension will be 14x14x6.
- Similarly, the third layer also involves in a convolution operation with 16 filters of size 5×5 followed by a fourth pooling layer with similar filter size of 2×2 and stride of 2. Thus, the resulting image dimension will be reduced to 5x5x16.
- Once the image dimension is reduced, the fifth layer is a fully connected convolutional layer with 120 filters each of size 5×5. In this layer, each of the 120 units in this layer will be connected to the 400 (5x5x16) units from the previous layers. The sixth layer is also a fully connected layer with 84 units.
- The final seventh layer will be a softmax output layer with 'n' possible classes depending upon the number of classes in the dataset.

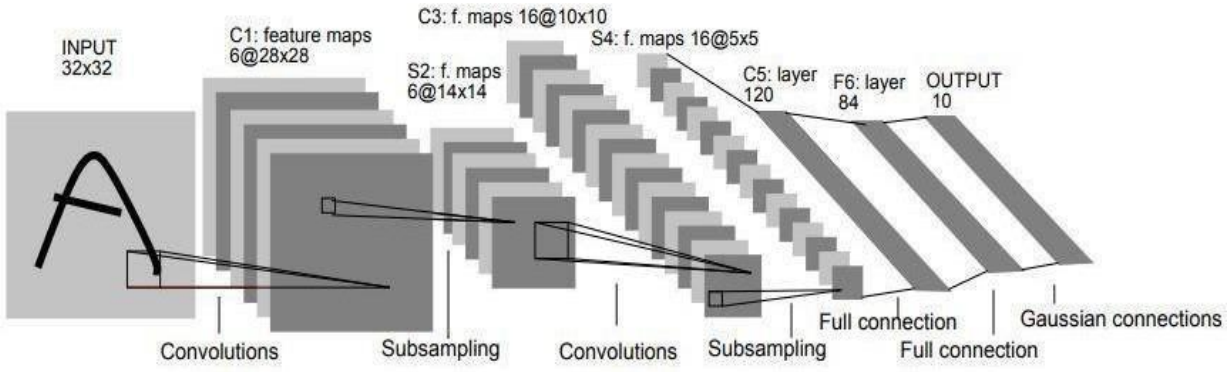


Figure 4.8: The above diagram is a representation of the 7 layers of the LeNet-5 CNN Architecture.

- This project is hostile to face spoof is measured as the classification of two-class issues. The two classes are genuine face class and ridiculed class of face. While preparation stage predicts CNN model class for preparing pictures, figures the unmitigated cross- entropy misfortune, and finally update the loads of system utilizing inclination plummetstrategy by back-engendering of slope for misfortune work.
- Here the use of color image blending technique that makes the classifier very excellent outcomes while spoofing. In every age, these loads utilizing preparing pictures are exploited to create the class scores and classification precision over approval pictures. After the completion of the work, the educated loads relating to most elevated accuracy is utilized for testing the face database.
- One of the main parts of Neural Networks is **Convolutional neural networks** (CNN). CNNs use image recognition and classification in order to detect objects, recognize faces, etc. They are made up of neurons with learnable weights and biases. Each specificneuron receives numerous inputs and then takes a weighted sum over them, where it passes it through an activation function and responds back with an output.
- CNNs are primarily used to classify images, cluster them by similarities, and then perform object recognition. Many algorithms using CNNs can identify faces, street signs, animals, etc.

How do CNNs work?

They are prompt by volume and utilize multi-channeled images. As opposed to flat images that humans can see that only have width and height, CNNs cannot recognize that. Due to digital color images having red-blue-green (RGB) encoding, CNNs mix those three colors to produce the color spectrum humans perceive.

SVM based Face Anti-spoofing

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

An SVM model is basically a representation of different classes in a hyper plane in multidimensional space. The hyper plane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyper plane (MMH).

The followings are important concepts in SVM −

- **Support Vectors** − Data points that are closest to the hyper plane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyper plane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyper plane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyper planes iteratively that segregates the classes in best way.
- Then, it will choose the hyper plane that separates the classes correctly.

Multiclass learning is referred as the tasks in which labels are assigned to instances where the labels are taken from a finite set of elements in supervised machine learning tasks. In existing system which is based on image distortion analysis different classifiers are used for different

spoof attacks. Each classifier is trained with combining all genuine samples and a single group of spoof samples.

The output from each of these classifiers has to be fused in order to get the finalresult. But fusion is a time consuming process and if output of any one of the classifiers is wrong it will affect the entire result. So in order to overcome this, in the proposed system a single multiclass SVM is used. A single multiclass SVM classifier is used for training and testing purpose which can also identify the type of spoof attack.

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier's detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

The support vector machines in scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. However, to usean SVM to make predictions for sparse data, it must have been fit on such data. For optimal

performance, use C-ordered numpy.ndarray (dense) or scipy.sparse.csr_matrix (sparse) with dtype=float64.

Advantages of SVM:

- Effective in high dimensional cases
- Its memory efficient as it uses a subset of training points in the decision function called support vectors
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

RESULTS AND DISCUSSION

CHAPTER - 5
RESULTS AND DISCUSSION

Replay-Mobile is a dataset for face recognition and presentation attack detection (anti-spoofing). The dataset consists of 1190 video clips of photo and video presentation attacks (spoofing attacks) to 40 clients, under different lighting conditions. These videos were recorded with an iPad Mini2 (running iOS) and a LG-G4 smartphone (running Android).

All videos have been captured using the front-camera of the mobile device (tablet or phone). The front-camera produces colour videos with a resolution of 720 pixels (width) by 1280pixels (height) and saved in ".mov" file-format. The frame rate is about 25 Hz. Real-accesses have been performed by the genuine user (presenting one's true face to the device). Attack- accesses have been performed by displaying a photo or a video recording of the attacked client, for at least 10 seconds.

Real client accesses have been recorded under five different lighting conditions (controlled, adverse, direct, lateral and diffuse). In addition, to produce the attacks, high- resolution photos and videos from each client were taken under conditions similar to those in their authentication sessions.

The 1190 real-accesses and attacks videos were then grouped in the following way:

- Training set: contains 120 real-accesses and 192 attacks under different lighting conditions;
- Development set: contains 160 real-accesses and 256 attacks under different lighting conditions;
- Test set: contains 110 real-accesses and 192 attacks under different lighting conditions;
- Enrollment set: contains 160 real-accesses under different lighting conditions, to be used **exclusively** for studying the baseline performance of face recognition systems. (This set is again partitioned into 'Training', 'Development' and 'Test' sets.)

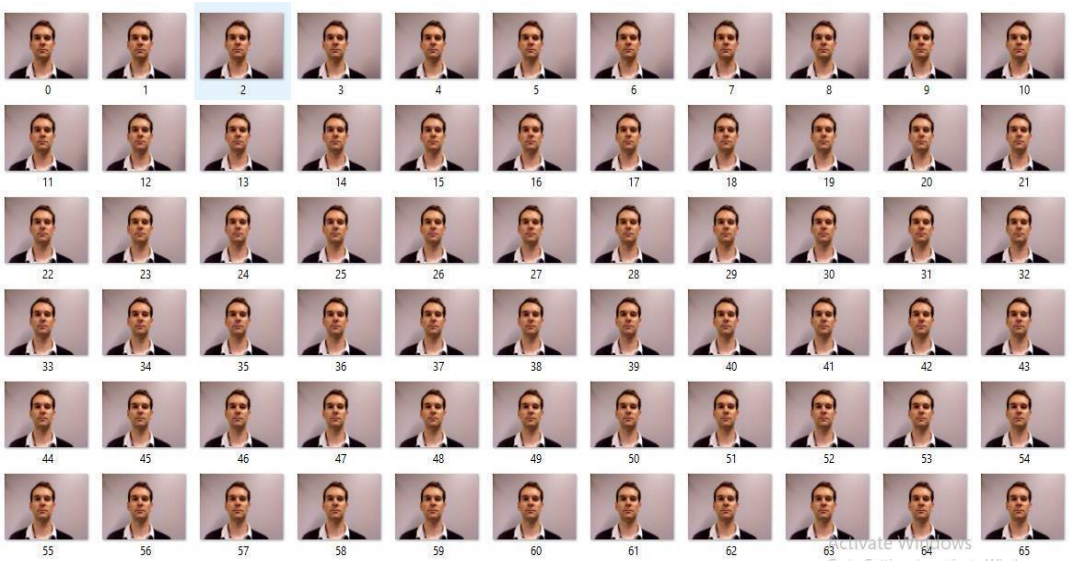


Figure 5.1: Each Real video is being converted into frames of 375

Figure 5.2: Each Spoofed video is being converted into frames of 230.

Gamma correction is also known as the *Power Law Transform.* First, our image pixel intensities must be scaled from the range *[0, 255]* to *[0, 1.0]*.

Where *I* is our input image and *G* is our gamma value. The output image *O* is then scaled back to the range *[0, 255]*.

Gamma values < *1* will shift the image towards the darker end of the spectrum while gamma values > *1* will make the image appear lighter. A gamma value of *G=1* will have no affect on the input image:

Figure 5.3: After Gamma correction

In image preprocessing using Local Binary Pattern, LBP is a pattern which is extracted from the image by processing its pixel in a specific logic format so that they change their value to binary values I.e. 0 & 1. LBP method provides the labeling of the pixels by finding the difference of neighborhood of each pixel and outputs that image area as the binary number.

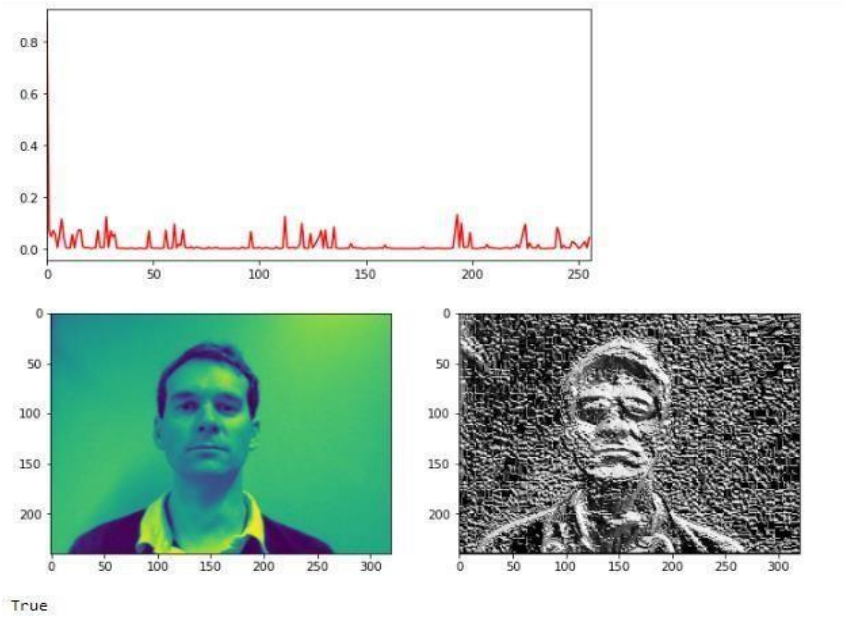


Figure 5.4: Facial features from the face using LBP texture descriptor.

Because of its high discrimination power and an ease and simplicity in computation, this operator has got a better popularity in its approach to be used in various applications. It has become a very unique approach for textual analysis other than the traditionally used textual analysis approach. The one of the most important property or feature of LBP is that it shows its robust behavior for the unicolor/bicolor in grey scale images.

The histogram is prepared on the basis of frequency distribution of lbp image formed from the actual image. This is done after the image is converted to LBP format and hence thisis done in mainly two ways: By taking the frequency of the pixel value and plotting on a histogram or by taking the probability of the frequency ofpixel value.

Figure 5.5: It will detect the Edges of a face.

Support vector machine (SVM) is supervised learning models have the ability of analyzing data for classification and regression purposes. SVM is supported by the theory of statistical learning, which has been developed by Vladimir. Although, there are many algorithms were developed for pattern classification, SVM is the best one among them.

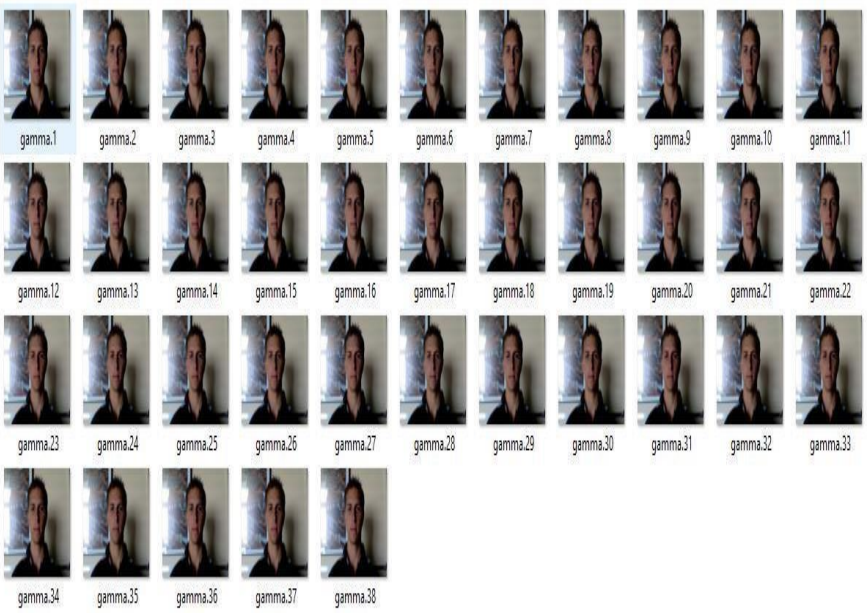


Figure 5.6: Training images after gamma correction

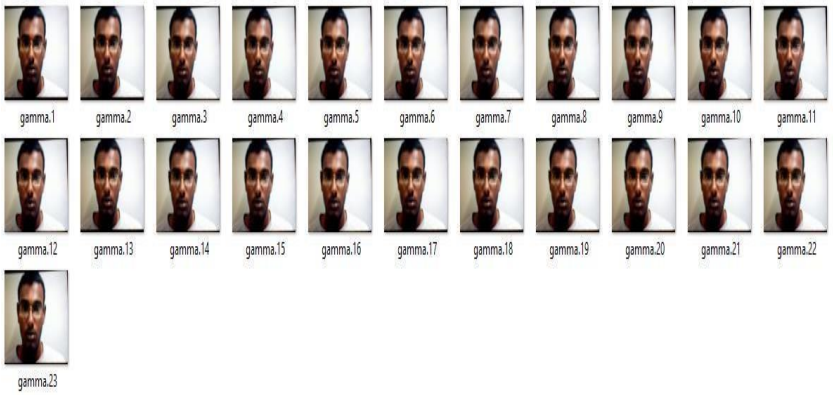


Figure 5.7: Testing images after gamma correction

The process of analyzing dataset items is based on their labels, where labels represent the class information of related data included in that datasets. Classification methods is implemented in many research fields such as bioscience, economy and forecasting because it support the decisions and summarize the collected data to be analyzed efficiently. Thus, classification methods are used by economists and doctors and not limited to computer scientists [2-4]. The

SVM principle is built on binary classification where it uses straight line to separates data points to classify the class label.

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known.

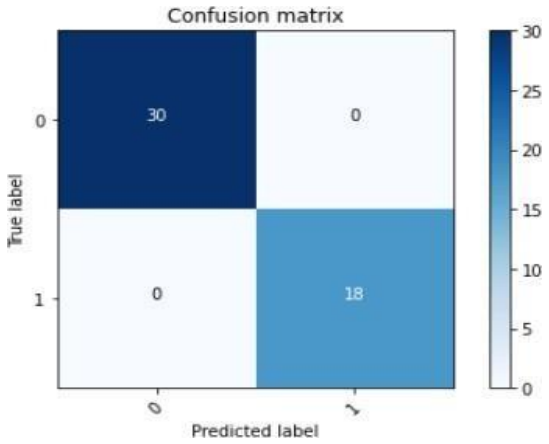


Figure 5.8: SVM Training model

The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2*2 tables, for 3 classes, it is 3*3 tables, and so on.
- The matrix is divided into two dimensions that are **predicted values** and **actual values** along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.

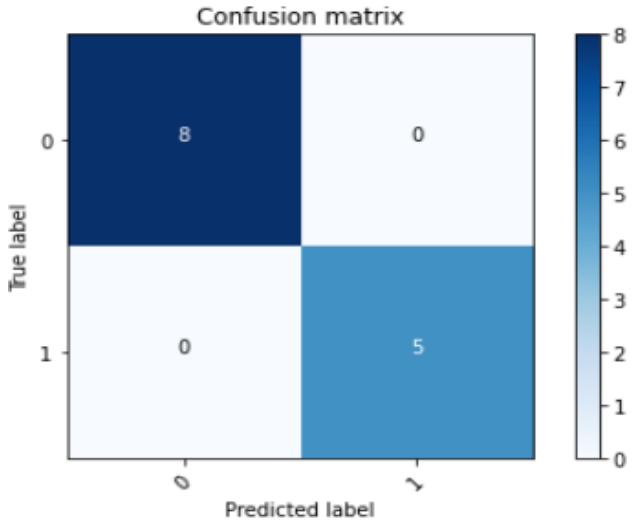


Figure 5.9: SVM Testing model

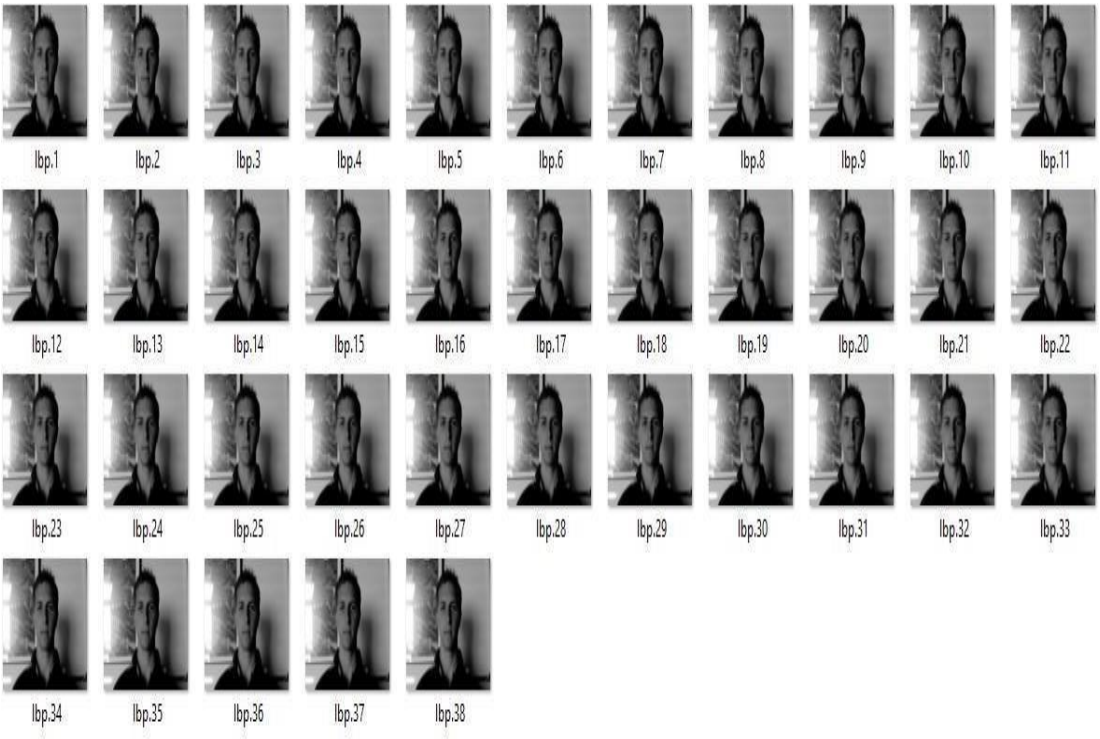


Figure 5.10: Result of LBP using Real non-attack image

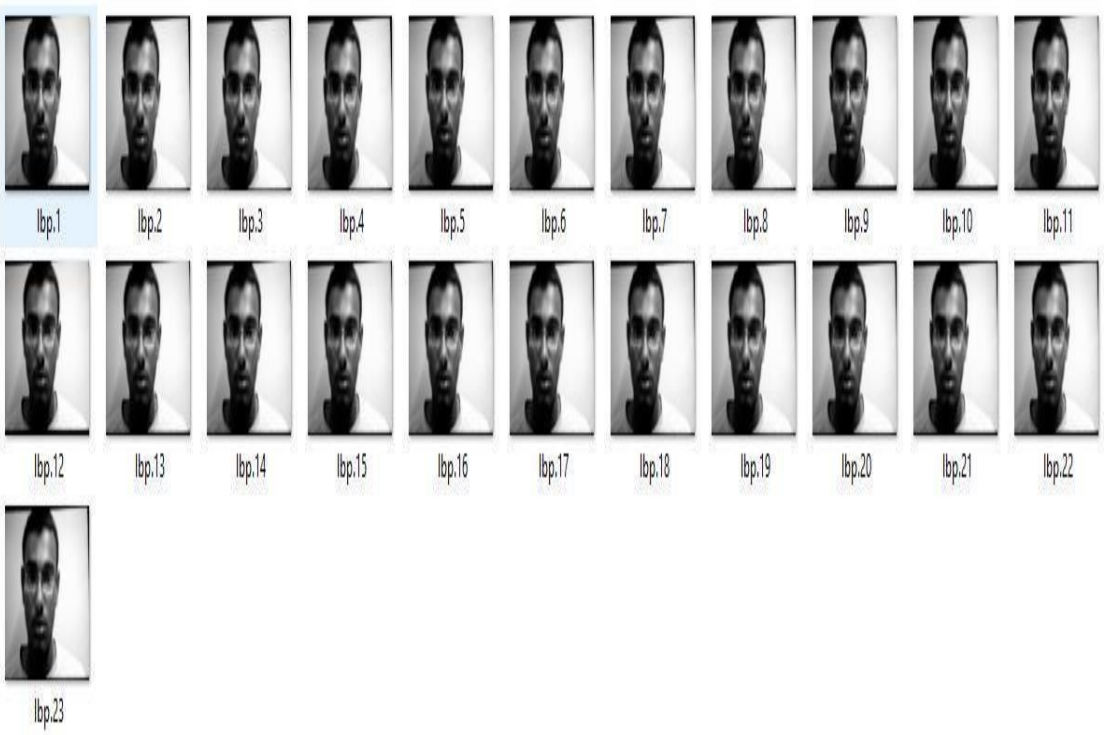


Figure 5.11: Result of LBP using Spoofed non-attack image

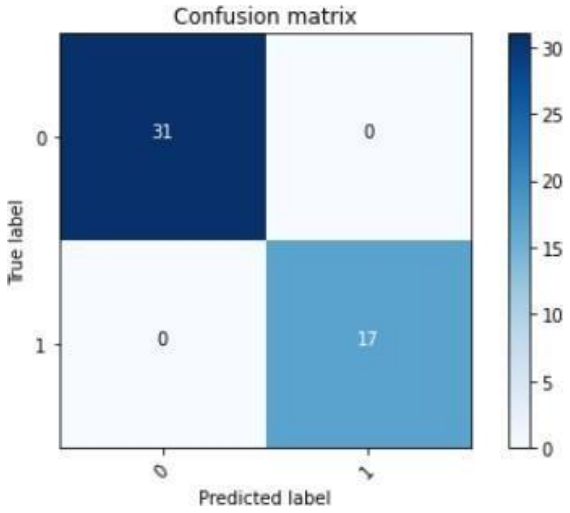


Figure 5.12: LBP Training model

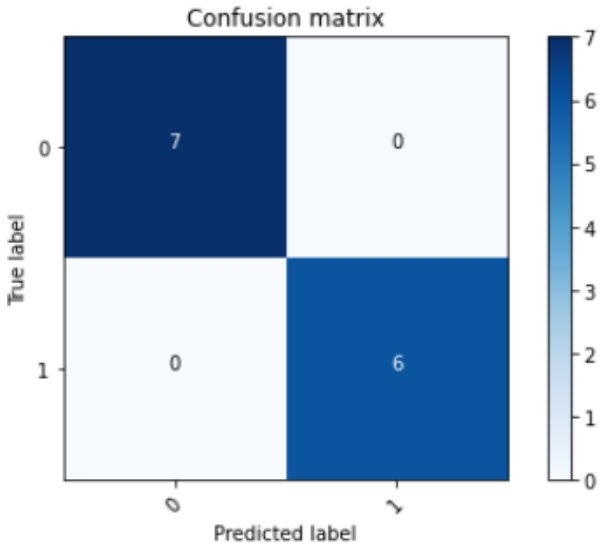


Figure 5.13: LBP Testing model

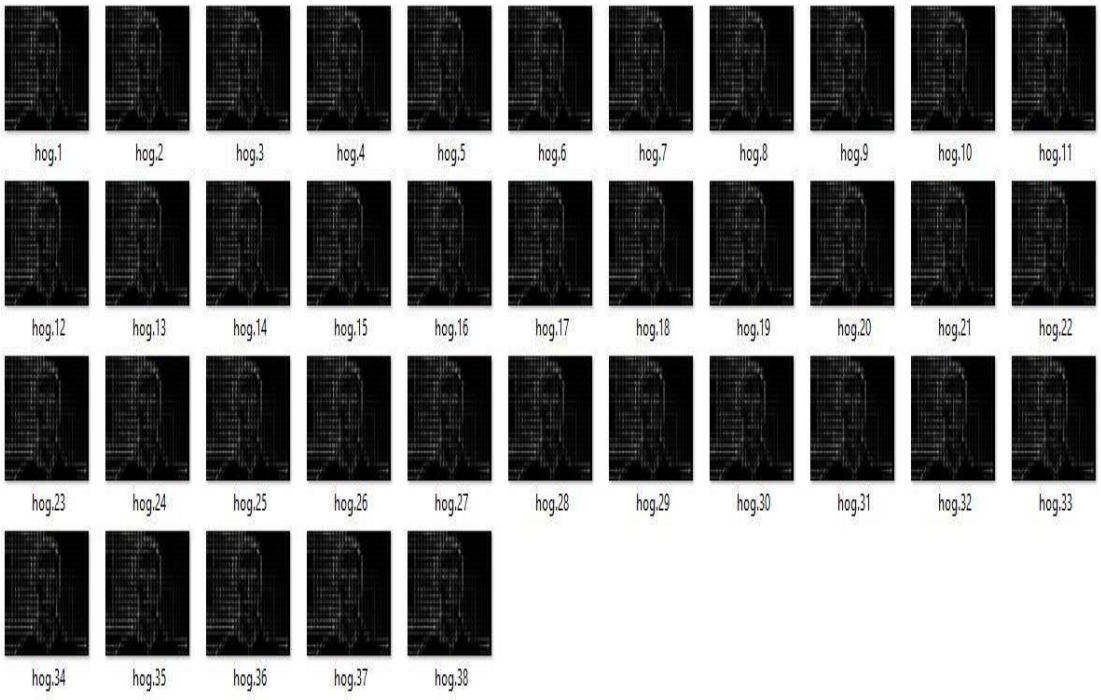


Figure 5.14: Result of HOG using Real non-attack image

Figure 5.15: Result of LBP using Spoofed non-attack image

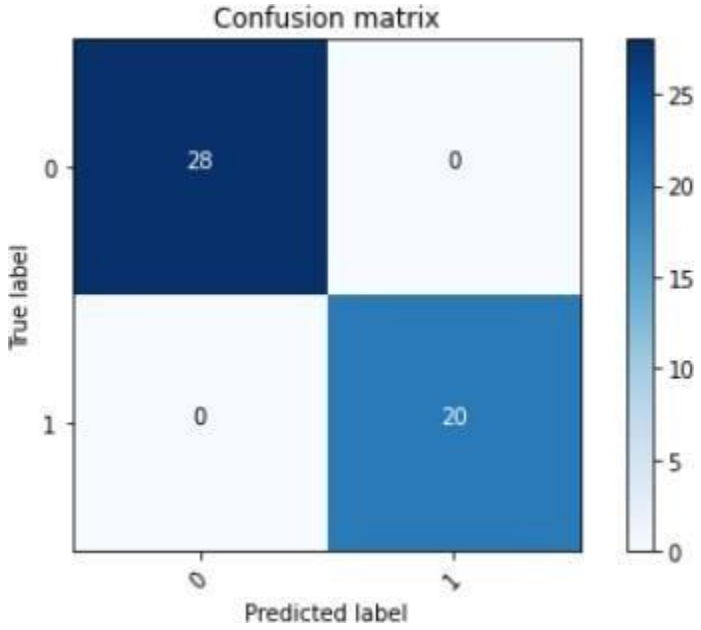


Figure 5.16: LBP Confusion Matrix for Training model

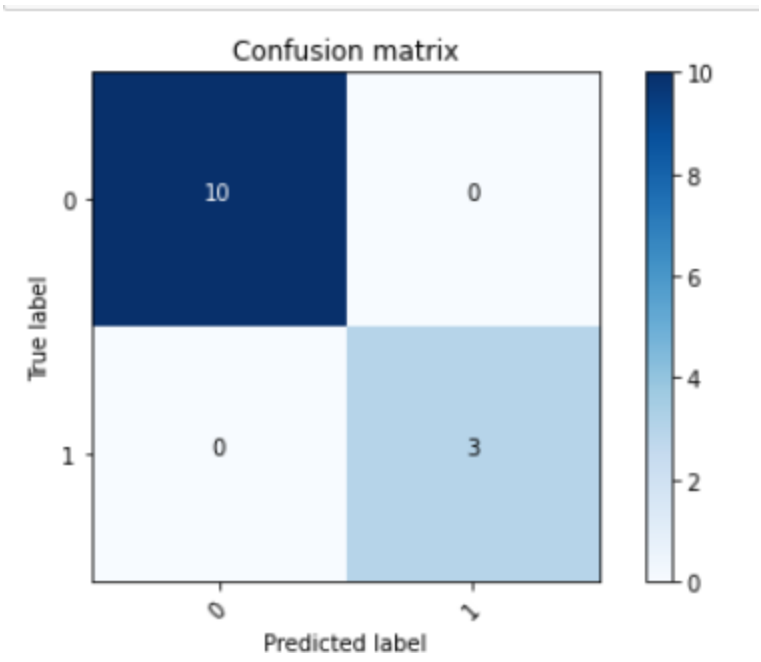


Figure 5.17: LBP Confusion Matrix for Testing model

CNN

```
Epoch 1/2
2/2 [==============================] - 35s 8s/step - loss: 0.8951 - accuracy: 0.6875 - val_loss: 0.5707 - val_accuracy: 1.0000
Epoch 2/2
2/2 [==============================] - 30s 8s/step - loss: 0.0825 - accuracy: 0.9792 - val_loss: 0.4660 - val_accuracy: 1.0000
```

Figure 5.18: Building CNN Model with 97.92% Accuracy

```
# Plot the Loss

plt.plot(history.history['loss'], label = 'train_loss')
plt.plot(history.history['val_loss'], label ='val loss')
plt.legend()
plt.show()
# plt.savefig('LossVal_loss')
```

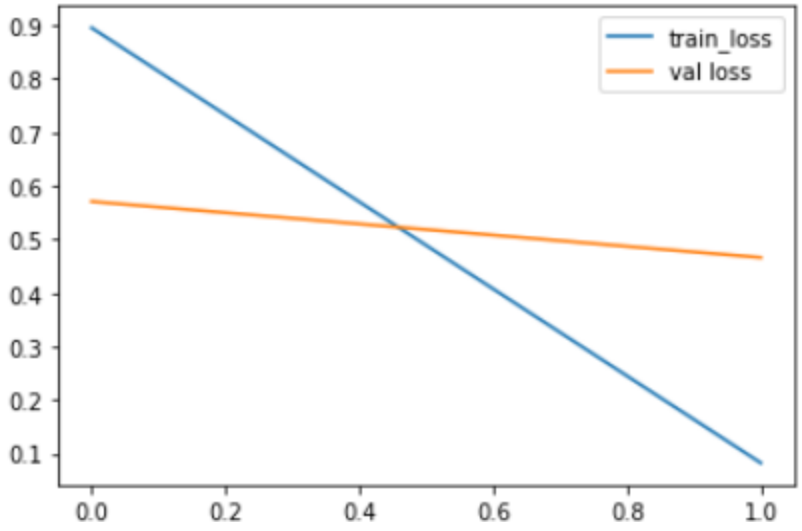


Figure 5.19: Loss Function for CNN Model

```
# Plot the Accuracy
plt.plot(history.history['accuracy'], label = 'train accuracy')
plt.plot(history.history['val_accuracy'], label ='val accuracy')
plt.legend()
plt.show()
# plt.savefig('valAccuracy')
```

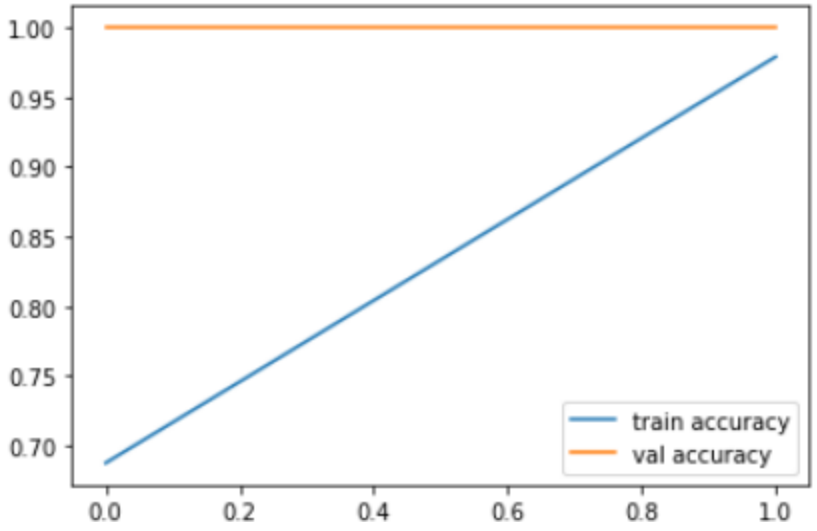


Figure 5.20: Accuracy for CNN model

SVM and CNN based Face Anti-spoofing

S.NO	METHODS	Accuracy
1.	Convolutional Neural Network	97.92%
2.	Support Vector Machine	100%

Figure 5.21: Performance Evaluation

CONCLUSION

CHAPTER - 6
CONCLUSION

Current face biometric systems are vulnerable to spoofing attacks. A spoofing attack occurs when a person tries to masquerade as someone else by falsifying data and thereby gaining illegitimate access. Inspired by image quality assessment, characterization of printing artifacts, and differences in light reflection, I propose to approach the problem of spoofing detection from texture analysis point of view. Indeed, face prints usually contain printing quality defects that can be well detected using texture and local shape features.

The proposed approach analyzes the texture and gradient structures of the facial images using a set of low-level feature descriptors, fast linear classification scheme and score level fusion. Compared to many previous works, the proposed approach is robust and does not require user-cooperation. In addition, the texture features that are used for spoofing detection can also be used for face recognition. This provides a unique feature space for coupling spoofing detection and face recognition.

This work proposes a new face anti- spoofing method based on color texture analysis. I analyze the joint color-texture information from the luminance and the chrominance channels is analyzed using a color local binary pattern descriptor. More specifically, the feature histograms are extracted from each image band separately.

The majority ofresearch on non-intrusive software-based face spoofing detection schemes has focused on the analysis of luminance information in face images, avoiding the Chroma component, which can be very useful for distinguishing fake faces from genuine ones. This paper explains a novel and appealing method to detect face spoofing across color texture analysis. By extracting complementary low-level feature descriptions from different color spaces, we exploit the joint color texture information from the luminance and chrominance channels. Add more algorithm to test the system .The best advantage it will suitable for both Machine Learning and Deep Learning.

- The proposed methodology using SVM provides greater accuracy. The reason behind 99.03% accuracy. Where the number of image is comparatively less. This results in 100% accuracy.
- CNN provides 97% accuracy with minimum loss. It is concluded that the SVM in machine learning and CNN in deep learning works better for face spoofing detection based on texture and quality of image.

REFERENCES

CHAPTER - 7

REFERENCES

[1] ZinelabidineBoulkenafet, JukkaKomulainen, AbdenourHadid Center for Machine Vision Research, University of Oulu, Finland: FACE ANTI-SPOOFING BASED ON COLOR TEXTURE ANALYSIS**(2018)**

[2] MdRezwan Hasan[1], S M Hasan Mahmud[2] and Xiang Yu Li[1] ,FACE ANTI-SPOOFING Using Texture-Based Techniques and Filtering Methods(2019).

[3] JukkaMa¨att ¨ a, AbdenourHadid, MattiPietik ¨ ainen Texture and quality analysis for face spoofing detection (29 January 2021)

[4] Kamlesh Kumar[1], Asif Ali Wagan[1*], MansoorAhmed Khuhro[1], Aamir Umrani[1], Ameen Chhajro[1], Abdul Hafeez[1], Asif Ali Laghari[1] Texture based Face recognition using GLCM and LBP schemes(2020)

[5] Chaorong Li, Huang Wei, Huafu Chen,LGLG-WPCA: An Effective Texture-based Method for Face Recognition (7 Jun 2019)

[6] Chaorong Li, Huang Wei, Huafu Chen, LGLG-WPCA: An Effective Texture-based Method for Face Recognition (2019).

[7] Balamurali K 1 , Chandru S 2 , MuhammedSohailRazvi 3 and V. Sathiesh Kumar, Face Spoof Detection Using VGG-Face Architecture(2021).

[8] Yaojie Liu, JeolStehouwer, Amin Jourabloo, Yousef Atoum, Xiaoming Liu,Face Anti-spoofing, Face Presentation Attack Detection(2018).

[9] JukkaMäättä; AbdenourHadid; MattiPietikäinen,Face spoofing detection from single images using micro-texture analysis(2011).

[10] Noor Al-Huda Taha a , Taha Mohammed Hassan b , Mohammed AkramYounis c, Face Spoofing Detection Using Deep CNN.

[11] Artur Costa-Pazo, Sushil Bhattacharjee, Esteban Vazquez-Fernandez and Sébastien Marcel,"The REPLAY-MOBILE Face Presentation-Attack Database", IEEE BIOSIG 2016.

ANNEXURE

CHAPTER - 8

ANNEXURE

```
import random
import numpy as np
import matplotlib.pyplot as plt
import math
from PIL import Image
import cv2 as cv
import glob
from skimage import color
from skimage.color import rgb2gray
from skimage.io import imread, imsave
from skimage.filters import threshold_otsu
from skimage import img_as_uint



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

#'''
for filepath in glob.iglob('C:/Users/Admin/Desktop/indhu/ind_pro/train/real/*'):
    num_in_folder=0

    if label>=108:
```

```
        path=filepath+'\*'
        print(path)
        for filefilepath in glob.iglob(filepath+'\*'):
            #if filefilepath[-1] == 'g':



                img = cv2.imread(filefilepath)
                img=cv2.resize(img,(400,300))

                img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                imgs.append([img,num_in_folder,label,img])
                #print(filefilepath)
                num_in_folder = num_in_folder+1

    label=label+1
print(filepath)
#'''

f1=random.choice(filepath)

filepath

print("total filepath",len(filepath))

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

```
random.shuffle(imgs)


test=[]


imgs, filepath
for file in filepath:
    i=0
    cap= cv.VideoCapture(filepath)
    while(cap.isOpened()):
        ret, frame = cap.read()
        if ret == False:
            break
        cv.imwrite('C:/Users/Admin/Desktop/indhu/ind_pro/new/'+str(i)+'.jpg',frame)


        #img_gray = color.rgb2gray(frame)
        #cv2.imwrite('C:/Users/Admin/ind_pro/grsc/frame'+str(i)+'.jpg',img_gray)
        i+=1
    cap.release()
    cv.destroyAllWindows()
#'''
for filepath in glob.iglob('C:/Users/Admin/Desktop/indhu/ind_pro/new/*'):
    num_in_folder=0


    if label>=108:
        path=filepath+'\*'
        print(path)
        for filefilepath in glob.iglob(filepath+'\*'):
            #if filefilepath[-1] == 'g':
```

```
            img = cv2.imread(filefilepath)
            img=cv2.resize(img,(400,300))

            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            imgs.append([img,num_in_folder,label,img])
            #print(filefilepath)
            num_in_folder = num_in_folder+1



'''
    for filefilepath in glob.iglob(filepath+'/R/*'):
        if filefilepath[-1] == 'g':



            img = cv2.imread(filefilepath)
            img=cv2.resize(img,(400,300))

            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            imgs.append([img,num_in_folder,label,img])
            print(filefilepath)
            num_in_folder = num_in_folder+1

        #'''
```

```
    label=label+1
print(filepath)
#"'
f1=random.choice(filepath)
filepath
print("total filepath",len(filepath))
img = cv.imread(filepath)

# METHOD 1: RGB
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
# compute gamma = log(mid*255)/log(mean)

mid = 0.5
mean = np.mean(gray)
gamma = math.log(mid*255)/math.log(mean)
print(gamma)
# do gamma correction
img_gamma1=np.power(img, gamma).clip(0,255).astype(np.uint8)
cv.imshow('gamma',img_gamma1)
cv.waitKey(0)
gamma_final=img_gamma1

# Display the images in subplots
cv.imwrite('savedimage.jpeg', img_gamma1)
cv.putText(img_gamma1,"g={}".format(gamma),(10,30),cv.FONT_HERSHEY_SIMX,
0.5, (0, 0, 255), 2)
#cv2.putText(img_gamma2,"g={}".format(gamma2),(10,30),cv2.FONT_HERSHEY_S
IMPLEX, 0.5, (0, 0, 255), 2)
```

```
cv.imshow("s2", np.hstack([img,img_gamma1]))
cv.waitKey(0)
#LBP


def lbp_basic(img_gamma1):
    basic_array = np.zeros(img_gamma1.shape,np.uint8)
    for i in range(basic_array.shape[0]-1):
        for j in range(basic_array.shape[1]-1):
            basic_array[i,j] = bin_to_decimal(cal_basic_lbp(img_gamma1,i,j))
    return basic_array
def cal_basic_lbp(img_gamma1,i,j):#Points larger than the center pixel are assigned a value of 1, and those smaller than the center pixel are assigned a value of 0. The binary sequence is returned
    sum = []
    if img_gamma1[i - 1, j ] > img_gamma1[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img_gamma1[i - 1, j+1 ] > img_gamma1[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img_gamma1[i , j + 1] > img_gamma1[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img_gamma1[i + 1, j+1 ] > img_gamma1[i, j]:
        sum.append(1)
```

```
    else:
        sum.append(0)
    if img_gamma1[i + 1, j ] > img_gamma1[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img_gamma1[i + 1, j - 1] > img_gamma1[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img_gamma1[i , j - 1] > img_gamma1[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img_gamma1[i - 1, j - 1] > img_gamma1[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    return sum
def bin_to_decimal(bin):#Binary to decimal
    res = 0
    bit_num = 0 #Shift left
    for i in bin[::-1]:
        res += i << bit_num # Shifting n bits to the left is equal to multiplying by 2 to the
nth power
        bit_num += 1
    return res
def show_basic_hist(a): #Draw histogram of original lbp
```

```
    hist = cv.calcHist([a],[0],None,[256],[0,256])
    hist = cv.normalize(hist,hist)
    plt.figure(figsize = (8,4))
    plt.plot(hist, color='r')
    plt.xlim([0,256])
    plt.show()
# do gamma correction
img_gamma1=np.power(img, gamma).clip(0,255).astype(np.uint8)
cv.imshow('LBP',img_gamma1)
cv.waitKey(0)
lbp=gamma_final
img1 = cv.cvtColor(img_gamma1,cv.COLOR_BGR2GRAY)
basic_array = lbp_basic(img1)
show_basic_hist(basic_array)
plt.figure(figsize=(11,11))
plt.subplot(1,2,1)
plt.imshow(img1)
plt.subplot(1,2,2)
plt.imshow(basic_array,cmap='Greys_r')
plt.show()
cv.imwrite('lbp1.jpg', img1)
#HOG
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
fig, axes = plt.subplots(1, 2, figsize=(8, 4))
```

```
#axes[0].imshow(real); axes[0].set_title("real")
axes[1].imshow(img1); axes[1]
[axe.axis('off') for axe in axes]
#plt.savefig("C:/Users/Admin/Desktop/indhu/ind_pro/demo.jpg", dpi=50)
cv.imwrite('C:/Users/Admin/Desktop/indhu/ind_pro/HOG.jpg', img1)
plt.tight_layout()
plt.show()
import matplotlib.pyplot as plt

from skimage.feature import hog
from skimage import data, exposure


#fd,hog_image=hog(spoof,orientations=8, pixels_per_cell=(16, 16),cells_per_block=(1,
1), visualize=True, channel_axis=-1)
fd, hog_image= hog(img1, orientations=8, pixels_per_cell=(12, 12),cells_per_block=(1,
1), visualize=True)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)
ax1.axis('off')
ax1.imshow(img1, cmap=plt.cm.gray)
ax1.set_title('Input image')

# Rescale histogram for better display
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))
ax2.axis('off')
ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
ax2.set_title('Histogram of Oriented Gradients')
plt.show()
```

```
cv.imwrite('HOG2.jpg', img1)
#SVM-GAMMA
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
Categories=['gamma_train', 'gamma_test']

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

datadir= 'C:/Users/Admin/Desktop/indhu/ind_pro/prepro1/gamma/'
#path which contains all the categories of images

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')
x_train.shape,x_test.shape,y_train.shape,y_test.shape
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn import svm
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import seaborn
%matplotlib inline
classifier = svm.SVC(kernel='linear')
classifier.fit(x_train, y_train)
prediction_SVM = classifier.predict(x_train)
prediction_SVM
class_names=np.array(['0','1']) # Binary label
import itertools
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
cm = confusion_matrix(y_train, prediction_SVM)
plot_confusion_matrix(cm,class_names)
#Testing the model
classifier = svm.SVC(kernel='linear')
classifier.fit(x_train,y_train)
prediction_SVM_all = classifier.predict(x_test)
cm = confusion_matrix(y_test, prediction_SVM_all)
plot_confusion_matrix(cm,class_names)
print('We have detected {} spoof / {} total spoof.'.format(str(cm[1][1]),str(cm[1][1]+cm[1][0])))
print('\nSo, the probability to detect a spoff is {}'.format(str(cm[1][1]/(cm[1][1]+cm[1][0]))))
```

```
print("the accuracy is : {}".format(str((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1])))))

#SVM-LBP
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
Categories=['lbp_train', 'lbp_test']

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

datadir= 'C:/Users/Admin/Desktop/indhu/ind_pro/prepro1/LBP/'
#path which contains all the categories of images

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')
x_train.shape,x_test.shape,y_train.shape,y_test.shape
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn import svm
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import seaborn
%matplotlib inline
classifier = svm.SVC(kernel='linear')
classifier.fit(x_train, y_train)
prediction_SVM = classifier.predict(x_train)
prediction_SVM
class_names=np.array(['0','1']) # Binary label
import itertools
def plot_confusion_matrix(cm, classes, title='Confusion matrix',cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
cm = confusion_matrix(y_train, prediction_SVM)
plot_confusion_matrix(cm,class_names)
print('We have detected {} spoof / {} total spoof.'.format(str(cm[1][1]),str(cm[1][1]+cm[1][0])))
print('\nSo, the probability to detect a spoff is {}'.format(str(cm[1][1]/(cm[1][1]+cm[1][0]))))
print("the accuracy is : {}".format(str((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1])))))
#Testing the model
classifier = svm.SVC(kernel='linear')
classifier.fit(x_train,y_train)
prediction_SVM_all = classifier.predict(x_test)
```

```
cm = confusion_matrix(y_test, prediction_SVM_all)
plot_confusion_matrix(cm,class_names)

#HOG-SVM
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
Categories=['train', 'test']

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

datadir= 'C:/Users/Admin/Desktop/indhu/ind_pro/prepro1/HOG/'
#path which contains all the categories of images

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')
x_train.shape,x_test.shape,y_train.shape,y_test.shape
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn import svm
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import seaborn
%matplotlib inline
classifier = svm.SVC(kernel='linear')
classifier.fit(x_train, y_train)
prediction_SVM = classifier.predict(x_train)
prediction_SVM
class_names=np.array(['0','1']) # Binary label
import itertools
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
```

```
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
cm = confusion_matrix(y_train, prediction_SVM)
plot_confusion_matrix(cm,class_names)
#Testing the model
classifier = svm.SVC(kernel='linear')
classifier.fit(x_train,y_train)
prediction_SVM_all = classifier.predict(x_test)
cm = confusion_matrix(y_test, prediction_SVM_all)
plot_confusion_matrix(cm,class_names)

#CNN
```

```
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],)
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_)
X_train[0].shape
epochs = 2
model = Sequential()
model.add(Conv1D(37, 2, activation='relu', input_shape = X_train[0].shape))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv1D(128, 2, activation='relu'))
model.add(BatchNormalization())
```

```
model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(10, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(optimizer=Adam(lr=0.000001), loss = 'binary_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test, y_test), verbose=1)
# Plot the Loss

plt.plot(history.history['loss'], label = 'train_loss')
plt.plot(history.history['val_loss'], label ='val loss')
plt.legend()
plt.show()
# plt.savefig('LossVal_loss')
# Plot the Accuracy
plt.plot(history.history['accuracy'], label = 'train accuracy')
plt.plot(history.history['val_accuracy'], label ='val accuracy')
plt.legend()
plt.show()
# plt.savefig('valAccuracy')
```