IMAGE FORGERY DETECTION

A Capstone Project report submitted
in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING
by

AFRAH NAAZ	**19K41A05B9**
TERALA HEMANTH	**19K41A0586**
NAGARAPU KRANTHI	**19K41A0550**
REDDYMASU BHAVANA	**19K41A0554**
BOLLA DEEKSHITH	**20K45A0512**

Under the guidance of
DR. P. CHAKRADHAR
Assistant Professor, Department of CSE.

SR Engineering College
Innovation . Creativity . Entrepreneurship

SR ENGINEERING COLLEGE

Ananthasagar, Warangal.

<u>CERTIFICATE</u>

This is to certify that this project entitled **“IMAGE FORGERY DETECTION**" is the bonafied work carried out by Afrah Naaz, Terala Hemanth, Nagarapu Kranthi, Reddymasu Bhavana, Bolla Deekshith as a Capstone Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING** during the academic year 2022-2023 under our guidance and Supervision.

Dr. P. Chakradhar	**Dr. M.Sheshikala**
Assisstant Professor,	Assoc. Prof. & HOD (CSE),
S R Engineering College,	S R Engineering College,
Ananthasagar, Warangal.	Ananthasagar, Warangal.

External Examiner

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide **Dr.P.Chakradhar, Asst.Prof.** as well as Head of the CSE Department **Dr. M.Sheshikala, Associate Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project co-ordinators **Mr. Sallauddin Md, Asst. Prof., and Y.Chanti Asst. Prof.** for their encouragement and support.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Principal, **Dr. V. MAHESH,** for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

Now-a-days biometric systems are useful in recognizing person's identity but criminals change their appearance in behaviour and psychological to deceive recognition system. To overcome from this problem we are using new technique called Deep Texture Features extraction from images and then building train machine learning model using CNN (Convolution Neural Networks) algorithm. This technique refer as LBPNet or NLBPNet as this technique heavily dependent on features extraction using LBP (Local Binary Pattern) algorithm.

In this project we are designing LBP Based machine learning Convolution Neural Network called LBPNET to detect fake face images. Here first we will extract LBP from images and then train LBP descriptor images with Convolution Neural Network to generate training model. Whenever we upload new test image then that test image will be applied on training model to detect whether test image contains fake image or non-fake image. Below we can see some details on LBP.

Local binary patterns (LBP) is a type of visual descriptor used for classification in computer vision and is a simple yet 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. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings.

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 counter-clockwise.

- 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 be processed using the Support vector machine, extreme learning machines, or some other machine learning algorithm to classify images. Such classifiers can be used for face recognition or texture analysis.

CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

ACRONYMS	ABBREVIATION
GAN	Generative Adversarial Network
DEEP FD	Deep Forgery Discriminator
RS	Remote Sensing
SAR	Synthetic Aperture Radar
SVM	Support Vector Machine
CNN	Convolutional Neural Network
KNN	K-nearest Neighbour
LBP	Local Binary Pattern
DFD	Data Flow Diagram
UML	Unified Modeling Language

1. INTRODUCTION

In this technological era a huge number of people have become victims of image forgery. A lot of people use technology to manipulate images and use it as evidences to mislead the court. So to put an end to this, all the images that are shared through social media should be categorized as real or fake accurately. Social media is a great platform to socialize, share and spread knowledge but if caution is not exercised, it can mislead people and even cause havoc due to unintentional false propaganda. While manipulation of most of the photo-shopped images is clearly evident due to pixelization & shoddy jobs by novices, some of them indeed appear genuine. Especially in the political arena, manipulated images can make or break a politician's credibility. Current forensic techniques require an expert to analyze the credibility of an image. We have implemented a system that can determine whether an image is fake or not with the help of machine learning and thereby making it available for the common public. This paper will unfold into three sections whereby first will focus on the second will focus on the Implementation details while the last part showcase the experimental result.

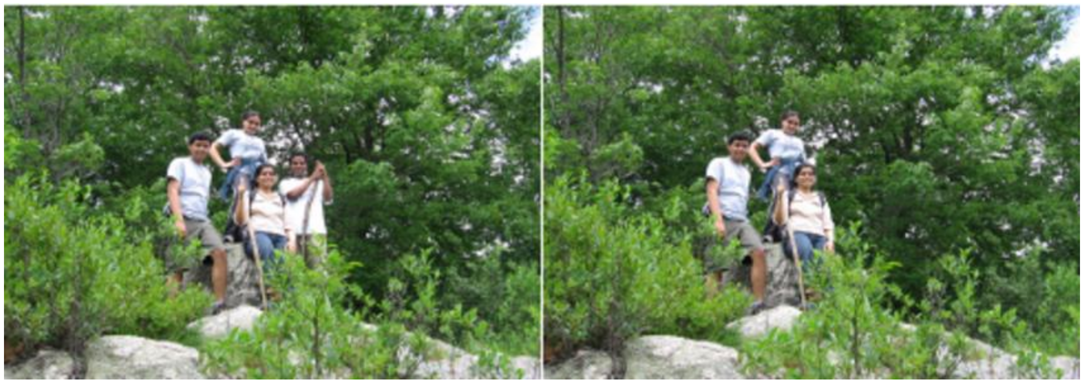


Fig-1: Tampered vs Original image

1.1. Image forgery classification

Image forgery detection aims to verify the authenticity of a digital image. Image authentication solution is classified into two types. (1) Active and (2) Blind or passive. An active forgery detection techniques, such as digital watermarking or digital signatures uses a known authentication code embedded into the image content before the images are sent through an unreliable public channel. By verifying he presence of such authentication code authentication may be proved by comparing with the original inserted code. However, this method requires special hardware or software to insert the authentication code inside the image before the image is being distributed. Passive or blind forgery detection technique uses the received image only for assessing its authenticity or integrity, without any signature or watermark of the original image from the sender. It is based on the assumption that although digital forgeries may leave no visual clues of having been tampered with, they may highly likely disturb the underlying statistics property or image consistency of a natural scene image which introduces new artifacts resulting in various forms of inconsistencies. These inconsistencies can be used to detect the forgery. This technique is popular as it does not need any prior information about the image. Existing techniques identify various traces of tampering and detect them separately with localization of tampered region. Still most of the image forgery techniques are remained unidentified and this articles objective is to explore all the existing blind forgery techniques and recent updates in this field.

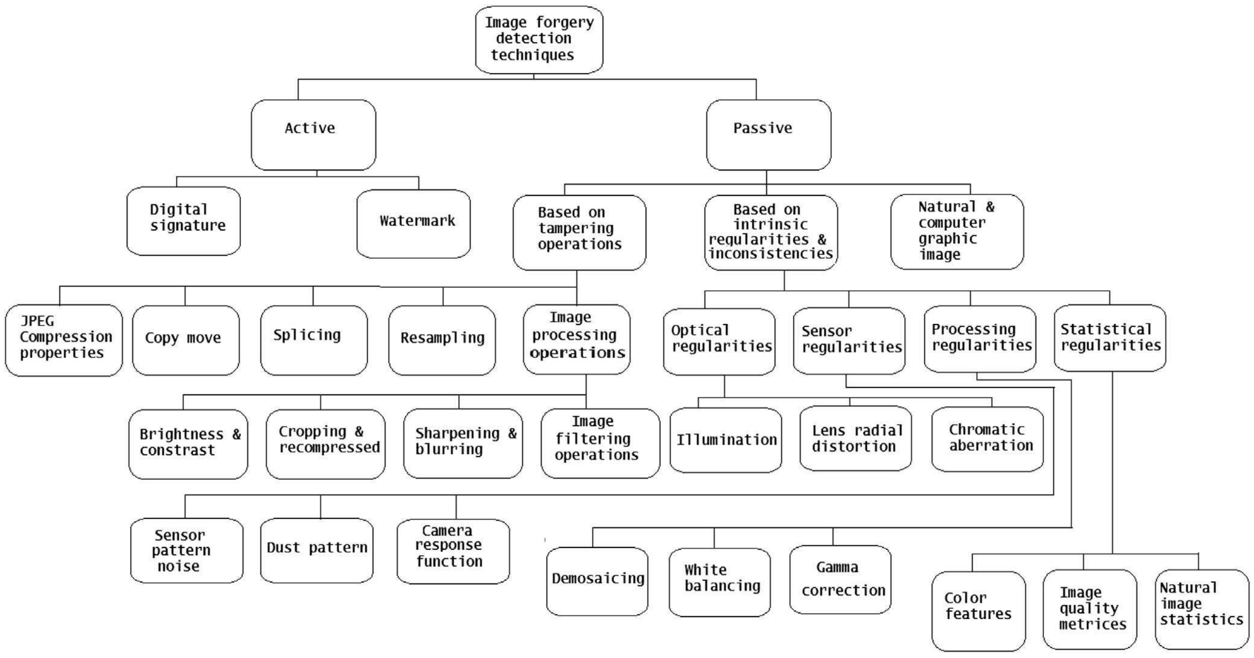


Fig-2: Digital image forgery detection techniques classification

2. LITERATURE SURVEY

[1] Learning to detect fake face images in the Wild.

Authors: Chih-Chung Hsu, Chia-Yen Lee, Yi-Xiu Zhuang

Although Generative Adversarial Network (GAN) can be used to generate the realistic image, improper use of these technologies brings hidden concerns. For example, GAN can be used to generate a tampered video for specific people and inappropriate events, creating images that are detrimental to a particular person, and may even affect that personal safety. In this paper, we will develop a deep forgery discriminator (DeepFD) to efficiently and effectively detect the computer-generated images. Directly learning a binary classifier is relatively tricky since it is hard to find the common discriminative features for judging the fake images generated from different GANs. To address this shortcoming, we adopt contrastive loss in seeking the typical features of the synthesized images generated by different GANs and follow by concatenating a classifier to detect such computer-generated images. Experimental results demonstrate that the proposed DeepFD successfully detected 94.7% fake images generated by several state-of-the-art GANs.

[2] SAR-to-Optical Image Translation Using Supervised Cycle-Consistent Adversarial Networks.

Authors: Xin Xu, Yue Yu, Lei Wang, Rui Yang

Optical remote sensing (RS) data suffer from the limitation of bad weather and cloud contamination, whereas synthetic aperture radar (SAR) can work under all weather conditions and overcome this disadvantage of optical RS data. However, due to the imaging mechanism of SAR and the speckle noise, untrained people are difficult to recognize the land cover types visually from SAR images. Inspired by the excellent image-to-image translation performance of Generative Adversarial Networks (GANs), a supervised Cycle-Consistent Adversarial Network (S-CycleGAN) was proposed to generate large optical images from the SAR images. When the optical RS data are unavailable or partly unavailable, the generated optical images can be alternative data that aid in land cover visual recognition for untrained people. The main steps of

SAR-to-optical image translation were as follows. First, the large SAR image was split to small patches. Then S-CycleGAN was used to translate the SAR patches to optical image patches. Finally, the optical image patches were stitched to generate the large optical image. A paired SAR-optical image dataset which covered 32 Chinese cities was published to evaluate the proposed method. The dataset was generated from Sentinel-1 (SEN-1) SAR images and Sentinel-2 (SEN-2) multi-spectral images. S-CycleGAN was applied to two experiments, which were SAR-to-optical image translation and cloud removal, and the results showed that S-CycleGAN could keep both the land cover and structure information well, and its performance was superior to some famous image-to-image translation models.

[3] Detection of GAN-Generated Fake Images over Social Networks.

Authors: Francesco Marra, Diego Gragnaniello, Davide Cozzolino, Luisa Verdoliva

The diffusion of fake images and videos on social networks is a fast growing problem. Commercial media editing tools allow anyone to remove, add, or clone people and objects, to generate fake images. Many techniques have been proposed to detect such conventional fakes, but new attacks emerge by the day. Image-to-image translation, based on generative adversarial networks (GANs), appears as one of the most dangerous, as it allows one to modify context and semantics of images in a very realistic way. In this paper, we study the performance of several image forgery detectors against image-to-image translation, both in ideal conditions, and in the presence of compression, routinely performed upon uploading on social networks. The study, carried out on a dataset of 36302 images, shows that detection accuracies up to 95% can be achieved by both conventional and deep learning detectors, but only the latter keep providing a high accuracy, up to 89%, on compressed data.

[4] Detecting Both Machine and Human Created Fake Face Images In the Wild.

Authors: Sangyup Lee, Shahroz Tariq, Hoyoung Kim, Youjin Shin

Due to the significant advancements in image processing and machine learning algorithms, it is much easier to create, edit, and produce high quality images. However, attackers can maliciously use these tools to create legitimate looking but fake images to harm others, bypass image detection algorithms, or fool image recognition classifiers. In this work, we propose neural network based classifiers to detect fake human faces created by both 1) machines and 2) humans. We use ensemble methods to detect GANs-created fake images and employ pre-processing techniques to improve fake face image detection created by humans. Our approaches focus on image contents for classification and do not use meta-data of images. Our preliminary results show that we can effectively detect both GANs-created images, and human-created fake images with 94% and 74.9% AUROC score.

[5] Stargan: Unified generative adversarial networks for multi-domain image-to-image translation.

Authors: Yunjey Choi, Minje Choi, Munyoung Kim and Jung-Woo Ha

Recent studies have shown remarkable success in image-to-image translation for two domains. However, existing approaches have limited scalability and robustness in handling more than two domains, since different models should be built independently for every pair of image domains. To address this limitation, we propose StarGAN, a novel and scalable approach that can perform image-to-image translations for multiple domains using only a single model. Such a unified model architecture of StarGAN allows simultaneous training of multiple datasets with different domains within a single network. This leads to StarGAN's superior quality of translated images compared to existing models as well as the novel capability of flexibly translating an input image to any desired target domain. We empirically demonstrate the effectiveness of our approach on a facial attribute transfer and a facial expression synthesis tasks.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

when capturing an image, additional required hidden information is associated with it for authentication and forgery protection purposes. Passive technique does not rely on extra information, but it analyzes some features extracted from the digital content of the image itself. Copy-move means coping a part of an image and pasting it into another place of the same picture whereas splicing is about taking a part of an image and pasting it into another

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

When comparing the visualization of features and the corresponding image patches, the latter has the greater variation since CNN mainly focuses on a discriminate structure.

3.2 PROPOSED SYSTEM:

In this project we are designing LBP Based machine learning Convolution Neural Network called LBPNET to detect fake face images. Here first we will extract LBP from images and then train LBP descriptor images with Convolution Neural Network to generate training model. Whenever we upload new test image then that test image will be applied on training model to detect whether test image contains fake image or non-fake image. Below we can see some details on LBP.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM:

CNN algorithm can predict images correctly up to 90% which is better prediction accuracy compare to all other algorithms such as SVM, KNN etc.

3.3 SYSTEM REQUIREMENTS:

3.3.1 HARDWARE REQUIREMENTS:

• System	: i3 Processor 2.4 GHz.
• Hard Disk	: 500GB.
• Ram	: 4GB.

3.3.2 SOFTWARE REQUIREMENTS:

- **Operating System:** Windows
- **Coding Language**: Python 3.7

3.4 SYSTEM STUDY

FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are

- ♦ ECONOMICAL FEASIBILITY
- ♦ TECHNICAL FEASIBILITY
- ♦ SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4. PROJECT DESIGN AND IMPLEMENTATION

4.1 SYSTEM ARCHITECTURE:

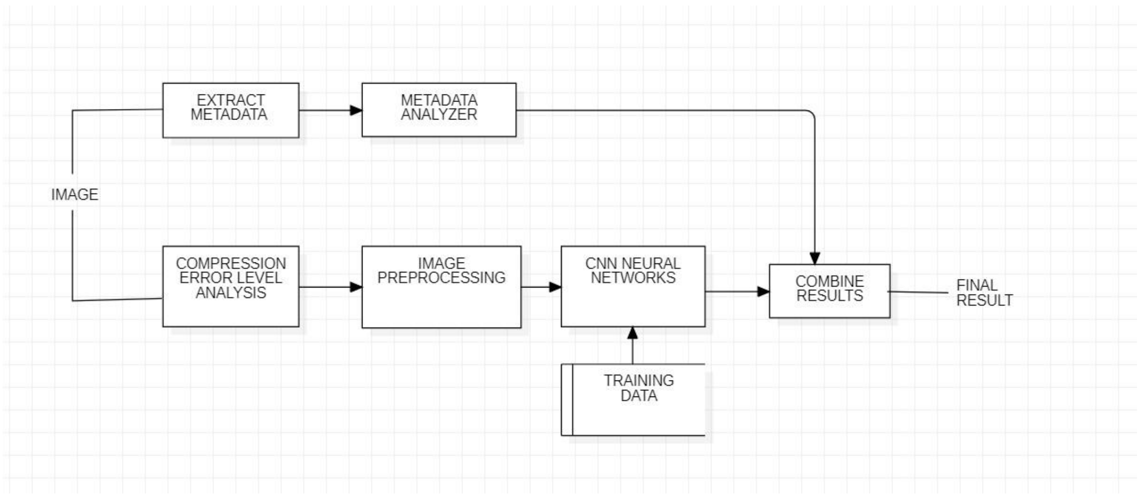


Fig-3: Architecture of image forgery detection

4.2 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

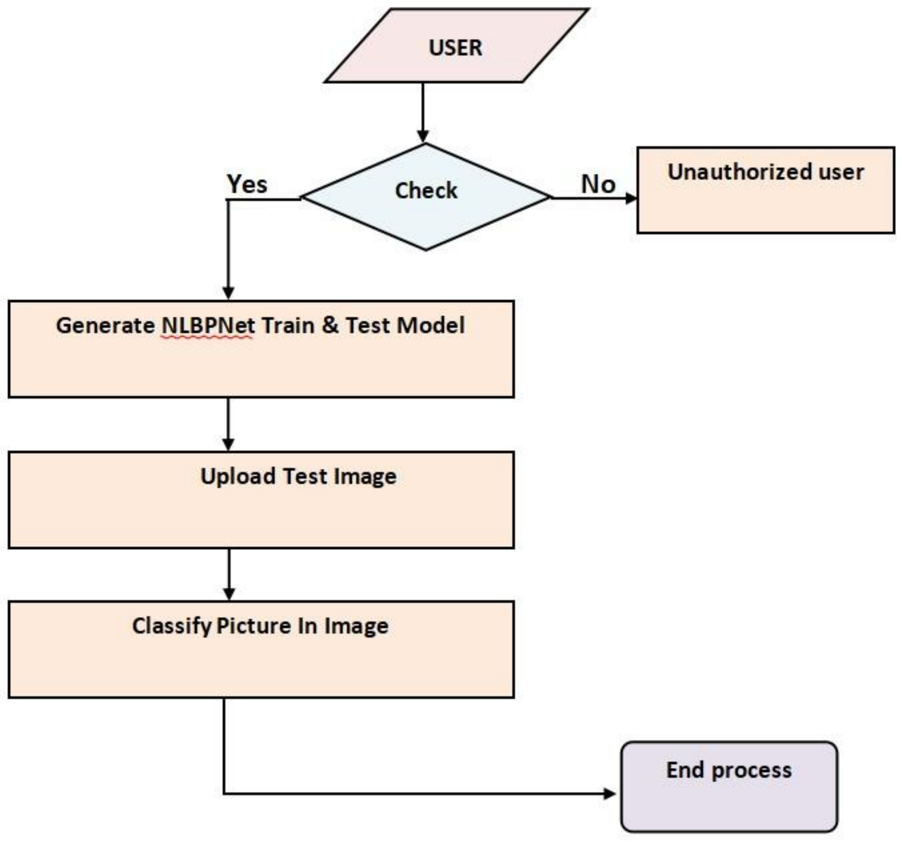

USER
Check
Yes
No
Unauthorized user
Generate NLBPNet Train & Test Model
Upload Test Image
Classify Picture In Image
End process

Fig - 4: Data flow diagram

4.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

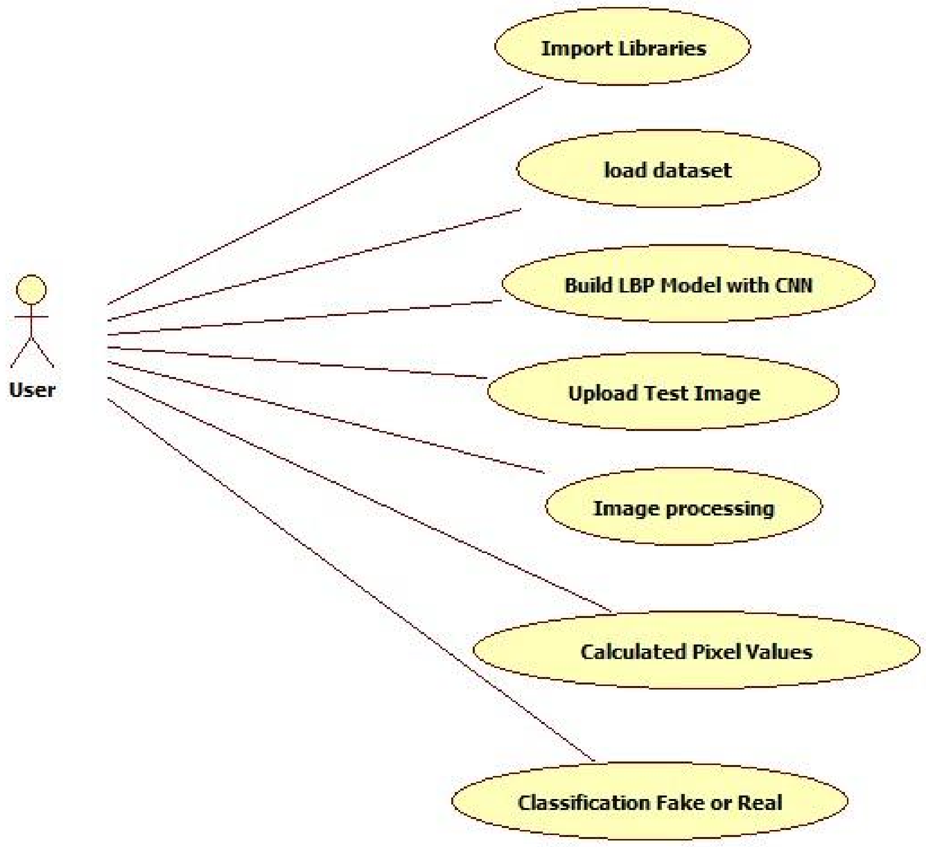


Fig - 5: Use Case diagram

4.3.2 CLASS DIAGRAM:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

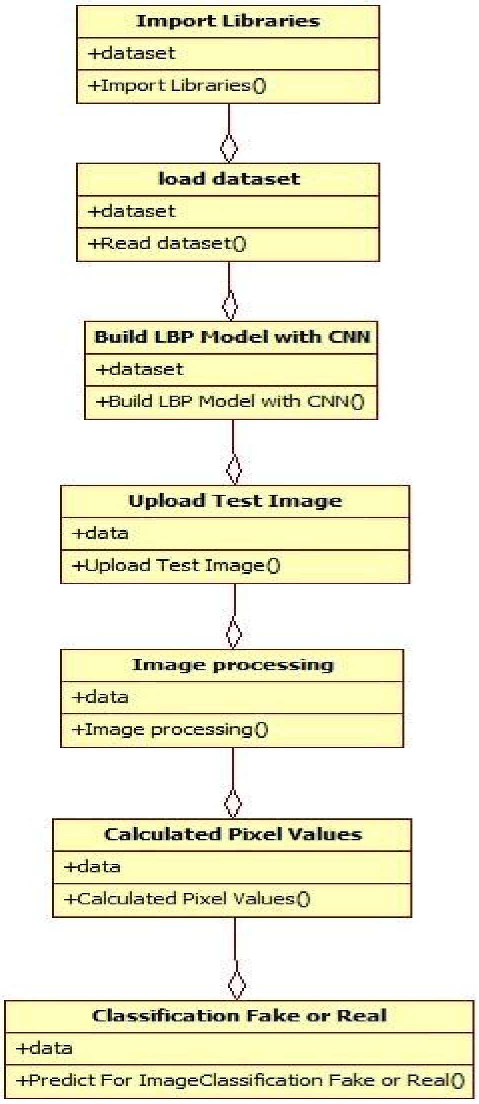


Fig - 6: Class diagram

4.3.3 OBJECT DIAGRAM:

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.

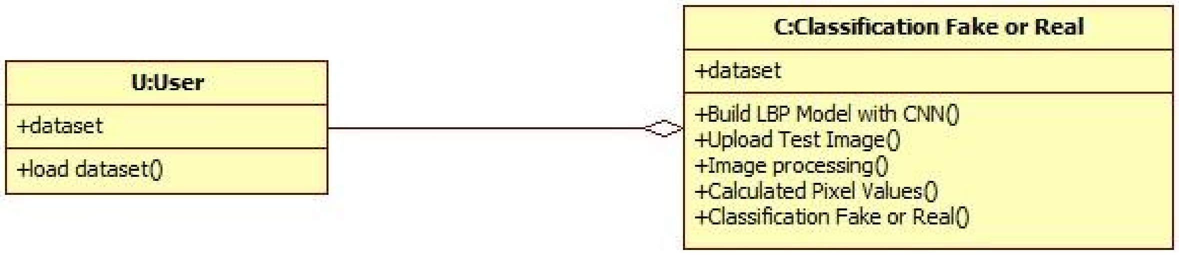

U:User
+dataset
+load dataset()
C:Classification Fake or Real
+dataset
+Build LBP Model with CNN()
+Upload Test Image()
+Image processing()
+Calculated Pixel Values()
+Classification Fake or Real()

Fig - 7: Object diagram

4.3.4 STATE DIAGRAM:

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

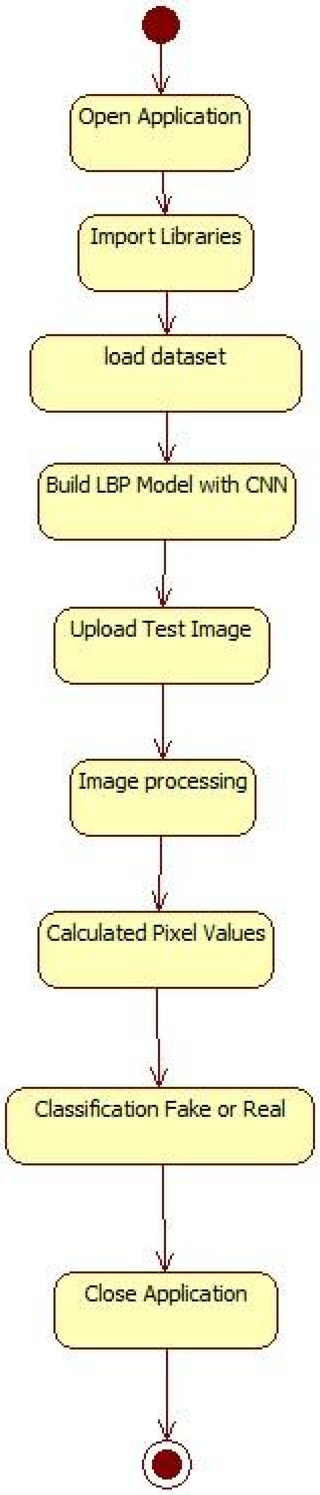


Fig - 8: State diagram

4.3.5 ACTIVITY DIAGRAM:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

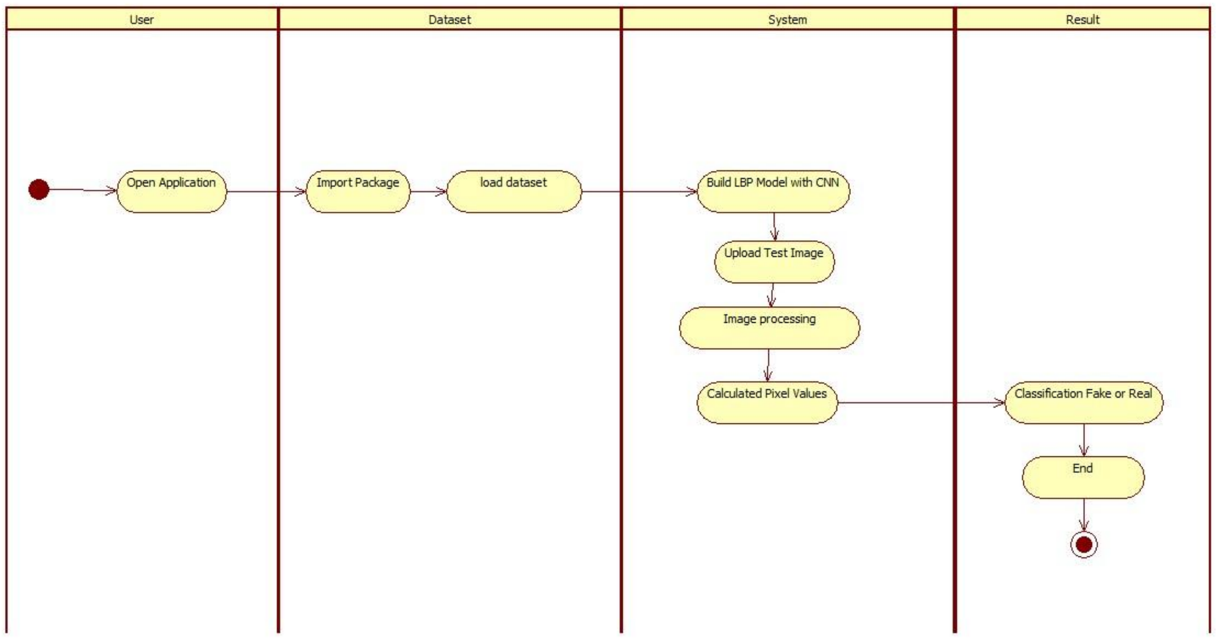


Fig - 9: Activity diagram

4.3.6 SEQUENCE DIAGRAM:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

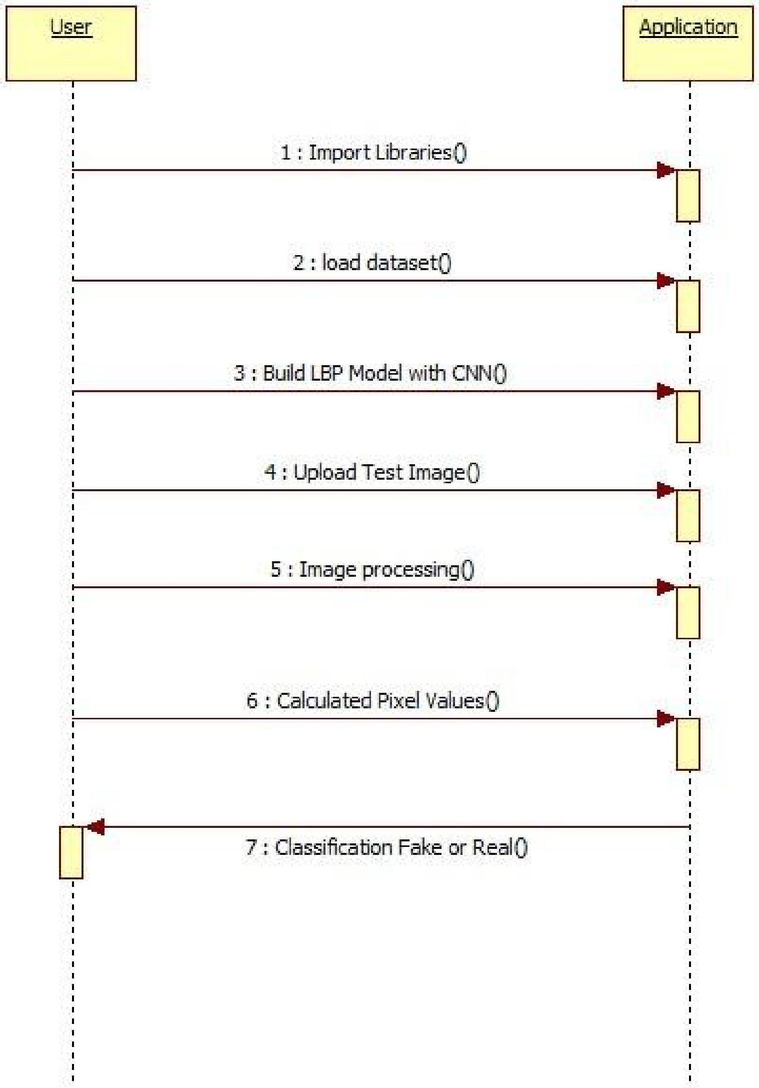


Fig - 10: Sequence diagram

4.3.7 COLLABORATION DIAGRAM:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

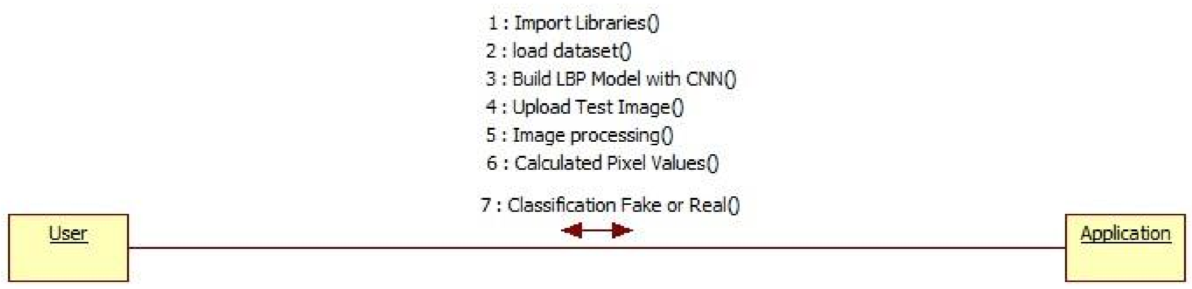


Fig - 11: Collaboration diagram

4.4 IMPLEMENTATION:

4.4.1 MODULES:

This project consists of following modules:

1) Generate NLBPNet Train & Test Model: in this module we will read all LBP images from LBP folder and then train CNN model with all those images.
2) Upload Test Image: In this module we will upload test image from ‘testimages’ folder. Application will read this image and then extract Deep Textures Features from this image using LBP algorithm.
3) Classify Picture In Image: This module apply test image on CNN train model to predict whether test image contains spoof or non-spoof face.

4.4.2 ALGORITHMS:

Local binary patterns (LBP) is a type of visual descriptor used for classification in computer vision and is a simple yet 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. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings.

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 counter-clockwise.
- ✧ 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 be processed using the Support vector machine, extreme learning machines, or some other machine learning algorithm to classify images. Such classifiers can be used for face recognition or texture analysis.

A useful extension to the original operator is the so-called uniform pattern, which can be used to reduce the length of the feature vector and implement a simple rotation invariant descriptor. This idea is motivated by the fact that some binary patterns occur more commonly in texture images than others. A local binary pattern is called uniform if the binary pattern contains at most two 0-1 or 1-0 transitions. For example, 00010000 (2 transitions) is a uniform pattern, but 01010100 (6 transitions) is not. In the computation of the LBP histogram, the histogram has a separate bin for every uniform pattern, and all non-uniform patterns are assigned to a single bin. Using uniform patterns, the length of the feature vector for a single cell reduces from 256 to 59. The 58 uniform binary patterns correspond to the integers 0, 1, 2, 3, 4, 6, 7, 8, 12, 14, 15, 16, 24, 28, 30, 31, 32, 48, 56, 60, 62, 63, 64, 96, 112, 120, 124, 126, 127, 128, 129, 131, 135, 143, 159, 191, 192, 193, 195, 199, 207, 223, 224, 225, 227, 231, 239, 240, 241, 243, 247, 248, 249, 251, 252, 253, 254 and 255.

4.4.3 CNN WORKING PROCEDURE

To demonstrate how to build a convolutional neural network based image classifier, we shall build a 6 layer neural network that will identify and separate one image from other. This network that we shall build is a very small network that we can run on a CPU as well. Traditional neural networks that are very good at doing image classification have many more parameters and take a lot of time if trained on normal CPU. However, our objective is to show how to build a real-world convolutional neural network using TENSORFLOW.

Neural Networks are essentially mathematical models to solve an optimization problem. They are made of neurons, the basic computation unit of neural networks. A neuron takes an input (say x), do some computation on it (say: multiply it with a variable w and adds another variable b) to produce a value (say; z= wx + b). This value is passed to a non-linear function called activation function (f) to produce the final output (activation) of a neuron. There are many kinds of activation functions. One of the popular activation function is Sigmoid. The neuron which uses sigmoid function as an activation function will be called sigmoid neuron. Depending on the activation functions, neurons are named and there are many kinds of them like RELU, TanH.

If you stack neurons in a single line, it's called a layer; which is the next building block of neural networks. See below image with layers

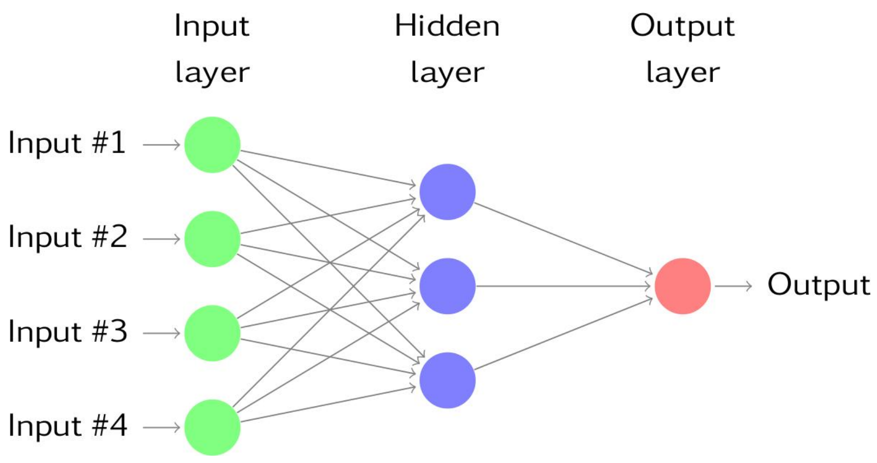


Fig - 12: CNN Networks

To predict image class multiple layers operate on each other to get best match layer and this process continues till no more improvement left.

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

MODULES USED IN PROJECT:

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object.
- Sophisticated (broadcasting) functions.
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

5. SYSTEM TESTING AND RESULTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Test Cases

USER REQUIREMENTS:

1. Home

Home:

Use case ID	Fake Image Identification
Use case Name	Home button
Description	Display home page of application
Primary actor	User
Precondition	User must open application
Post condition	Display the Home Page of an application
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	
Attachments	N/A

5.2 RESULTS AND ANALYSIS:

To run this project double click on ‘run.bat’ file to get below screen

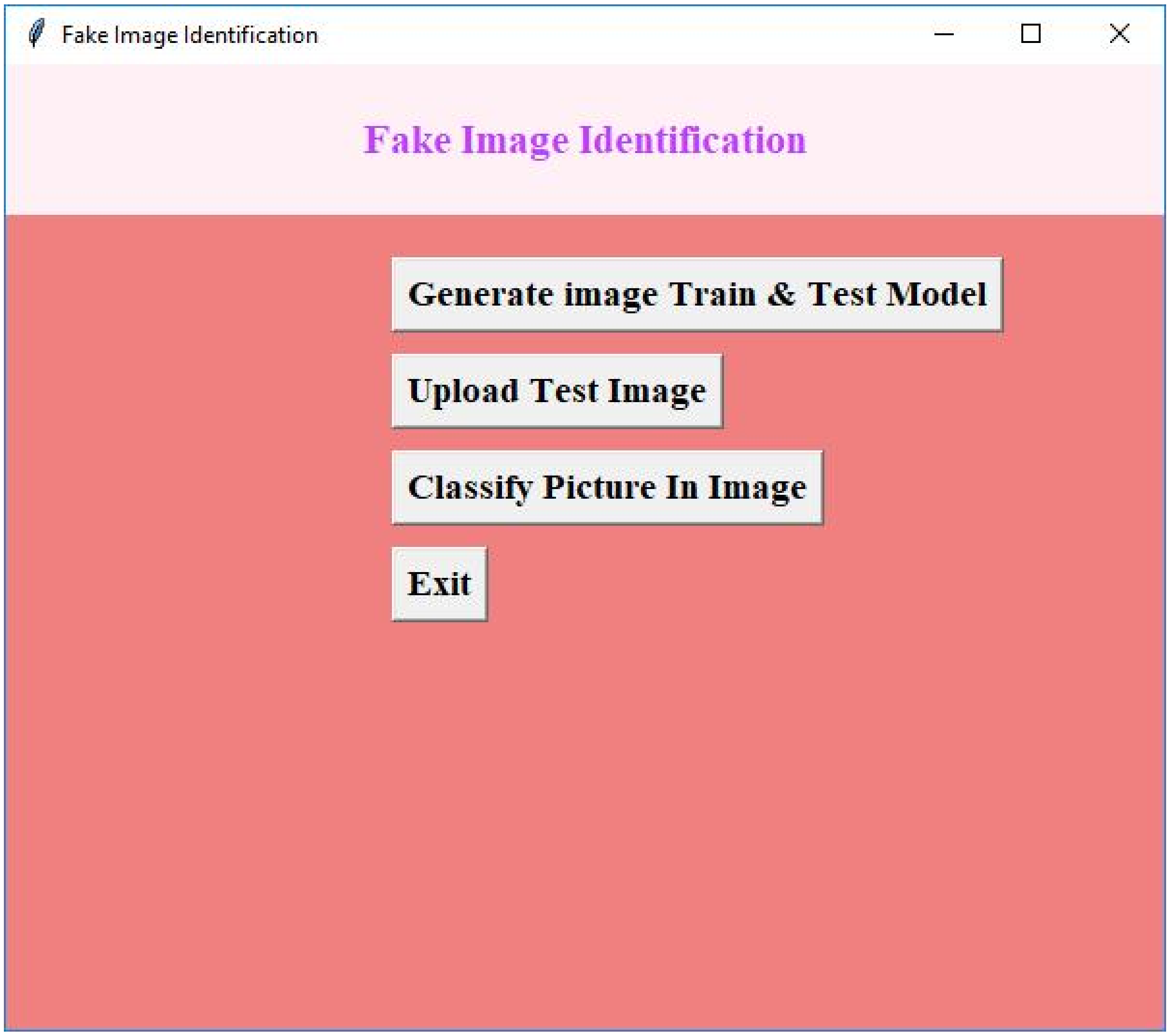
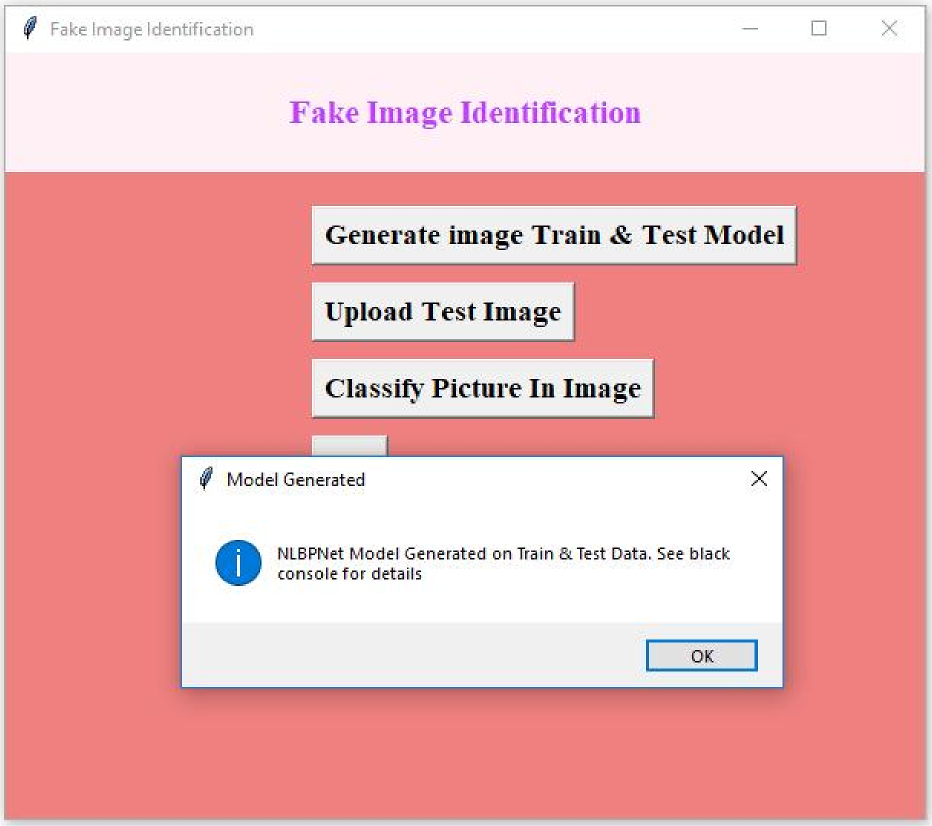

Fake Image Identification
Fake Image Identification
Generate image Train & Test Model
Upload Test Image
Classify Picture In Image
Exit

Fig - 13: Output screen 1

In the above screen click on ‘Generate Image Train & Test Model’ button to generate CNN model using LBP images contains inside LBP folder.



Fake Image Identification
Fake Image Identification
Generate image Train & Test Model
Upload Test Image
Classify Picture In Image
Model Generated
OK

NLBPNet Model Generated on Train & Test Data. See black console for details

Fig - 14: Output screen 2

In the above screen we can see CNN LBPNET model generated. Now click on 'Upload Test Image' button to upload test image

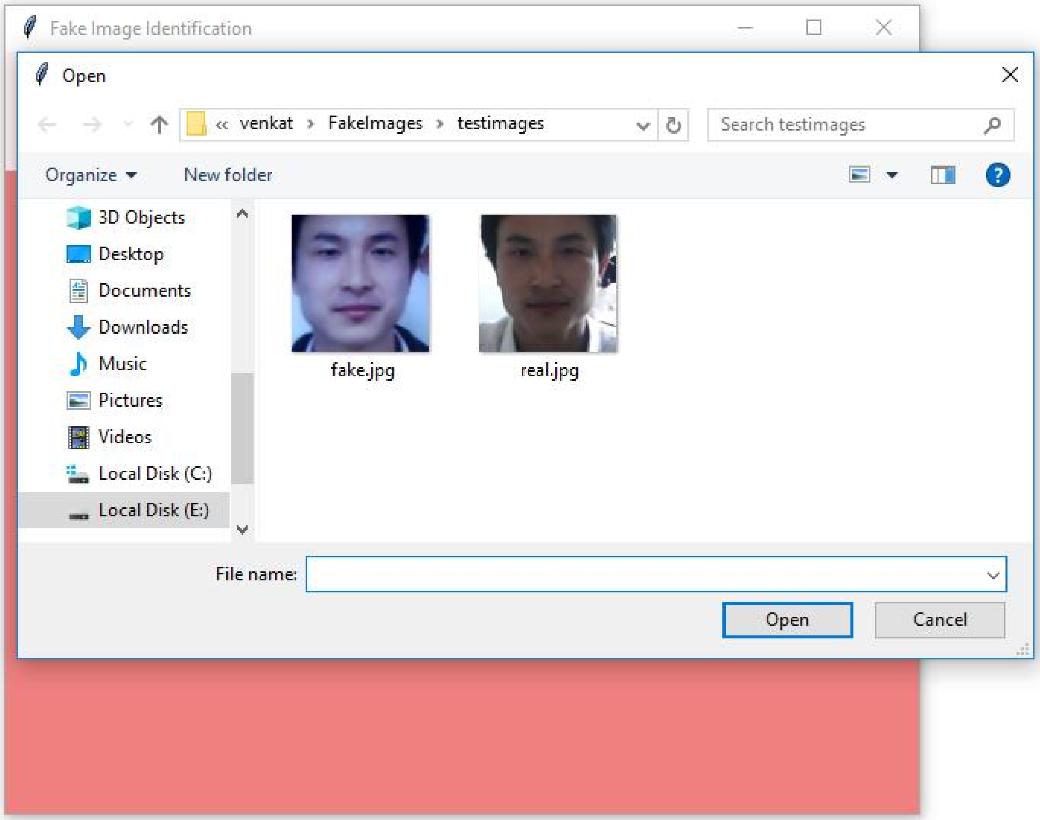


Fig - 15: Output screen 3

In the above screen we can see there are two faces from the same person but in different appearances. For simplicity we gave image name as fake and real to test whether application can detect it or not. In the above screen we are uploading fake image and then click on 'Classify Picture In Image' button to get below result.

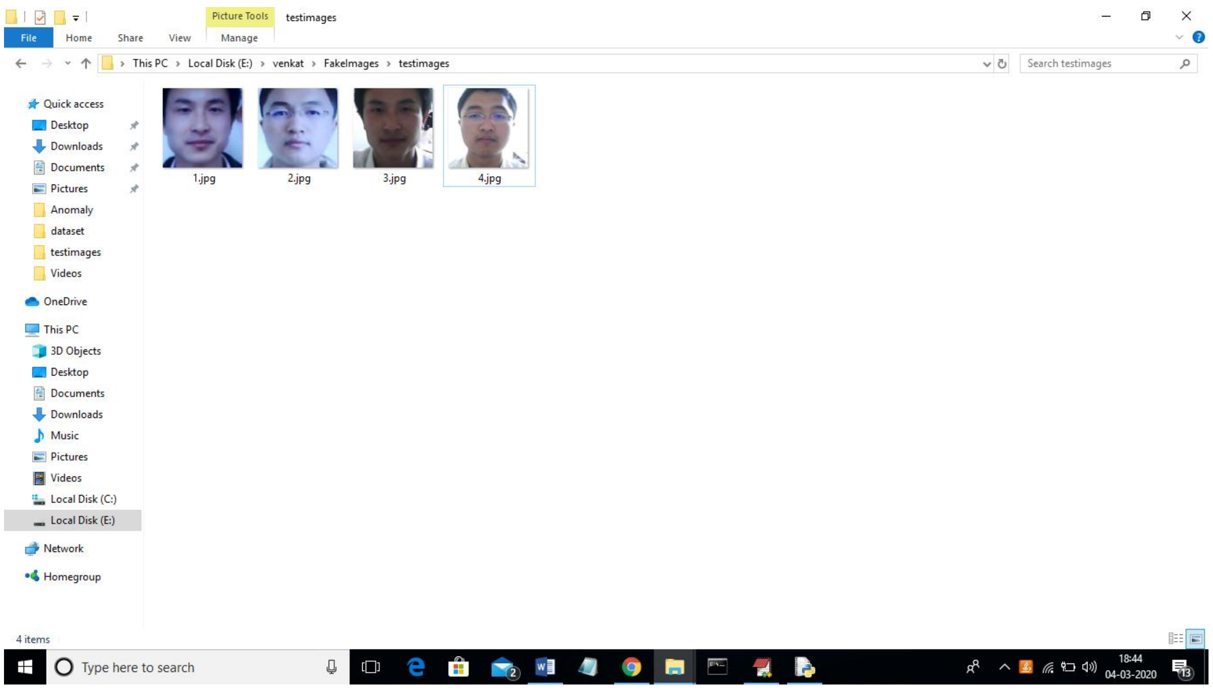


Fig - 16: Output screen 4

In the above screen we can see all the real faces will have normal light and in fake faces people will try some editing to avoid detection but this application will detect whether the picture is real or fake.

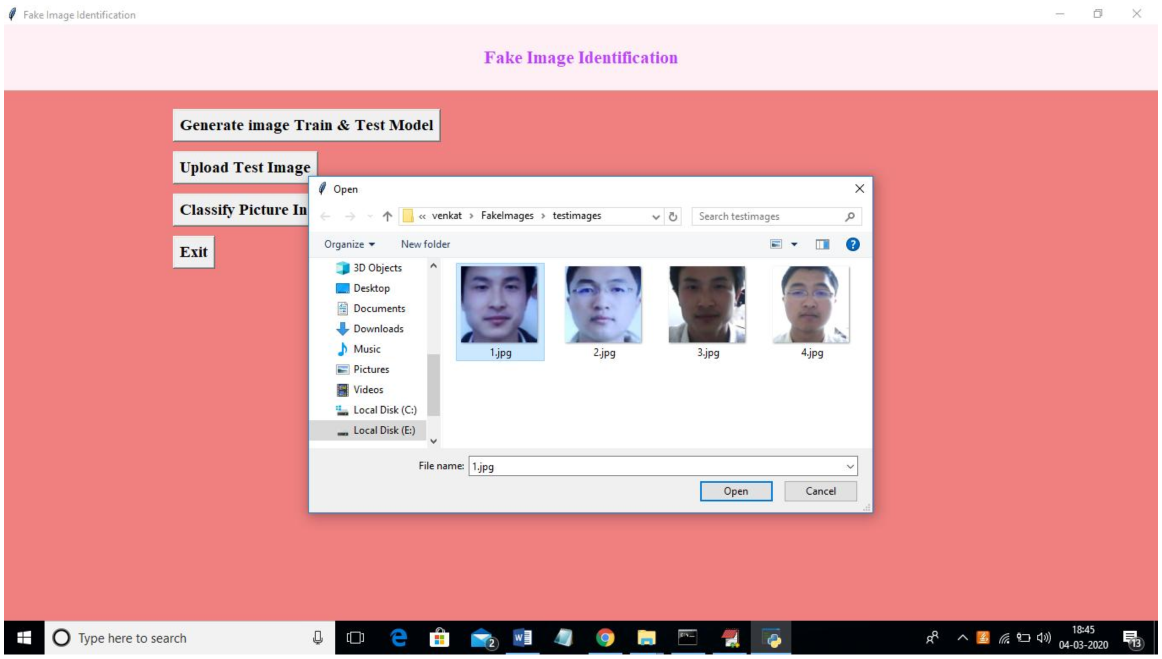


Fig - 17: Output screen 5

In the above screen we are uploading 1.jpg and after uploading click on open button to get below screen.

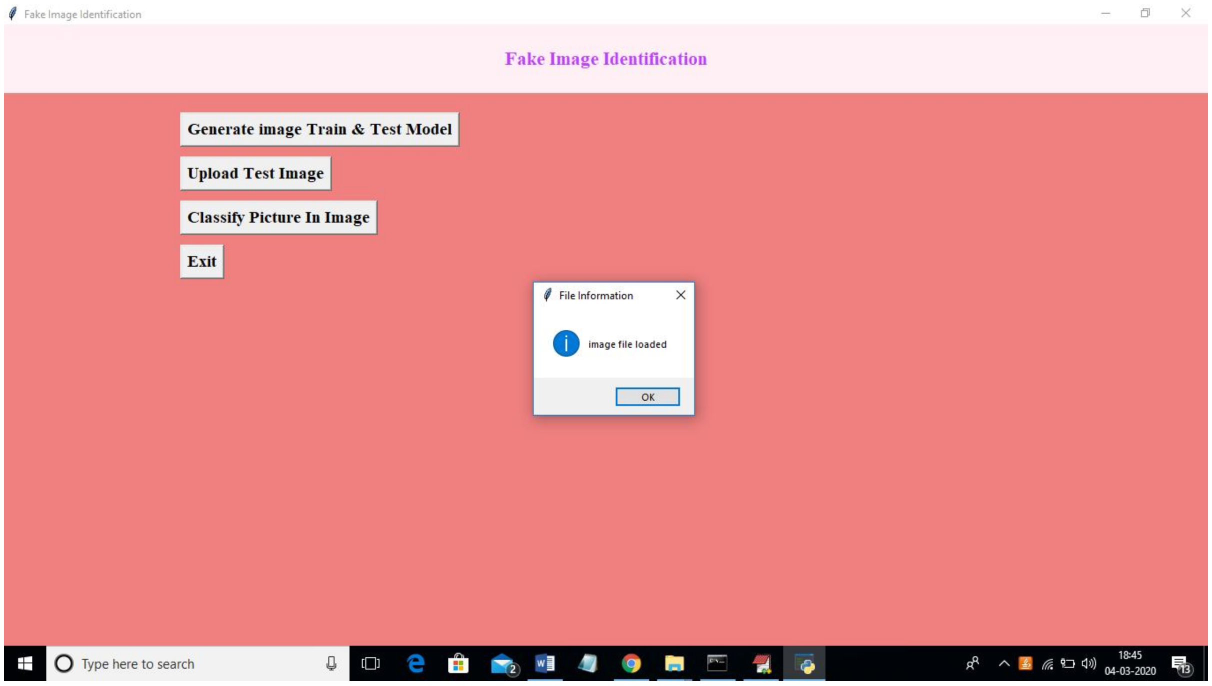

Fake Image Identification
Fake Image Identification
Generate image Train & Test Model
Upload Test Image
Classify Picture In Image
Exit
File Information
image file loaded
OK

Fig - 18: Output screen 6

And now click on 'classify Picture in Image' to get below details.

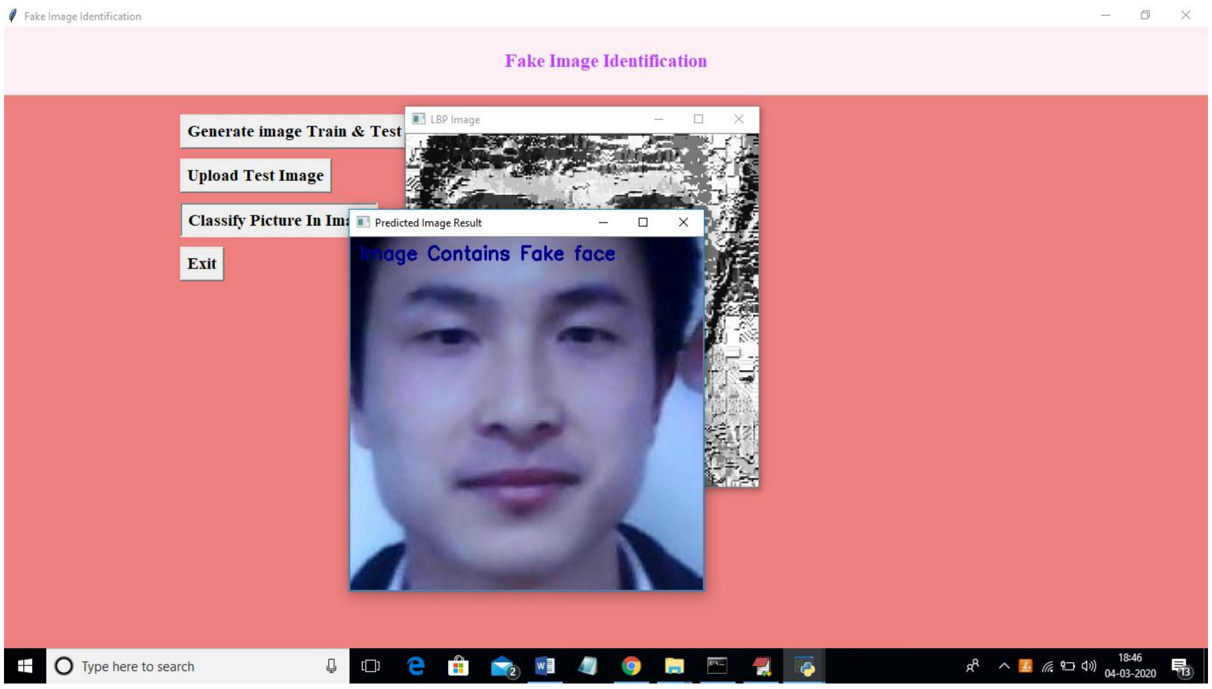


Image Contains Fake face

Fig - 19: Output screen 7

Now we are getting result as image contains Fake face. Similarly we can also try other images. We need to make CNN model with new images so it can detect those images too.

6. CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION:

In this paper, we have proposed a novel common fake feature network based the pairwise learning, to detect the fake face/general images generated by state-of-the-art GANs successfully. The proposed CFFN can be used to learn the middle- and high-level and discriminative fake feature by aggregating the cross-layer feature representations into the last fully connected layers. The proposed pairwise learning can be used to improve the performance of fake image detection further. With the proposed pairwise learning, the proposed fake image detector should be able to have the ability to identify the fake image generated by a new GAN. Our experimental results demonstrated that the proposed method outperforms other state-of-the-art schemes in terms of precision and recall rate.

6.2 FUTURE SCOPE:

Future Enhancement is being planned to further analyze and enhance the protocol to improve new feature in fake image detection. The proposed pairwise learning can be used to improve the performance of fake image detection further. With the proposed pairwise learning.

7. REFERENCES

1. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. arXiv Preprint, arXiv:1710.10196 2017. 256

2. Brock, A.; Donahue, J.; Simonyan, K. Large scale gan training for high fidelity natural image synthesis. arXiv Preprint, arXiv:1809.11096 2018.

3. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent 259 adversarial networks. arXiv Preprint, 2017.

4. AI can now create fake porn, making revenge porn even more complicated,. http://theconversation.com/ai-can-now-create-fake-porn-making-revenge-porn-even-more-complicated-92267, 262 2018.

5. Hsu, C.; Lee, C.; Zhuang, Y. Learning to detect fake face images in the Wild. 2018 International Symposium 264 on Computer, Consumer and Control (IS3C), 2018, pp. 388–391. doi:10.1109/IS3C.2018.00104.

6. H.T. Chang, C.C. Hsu, C.Y.a.D.S. Image authentication with tampering localization based on watermark 266 embedding in wavelet domain. Optical Engineering 2009, 48, 057002.

7. Hsu, C.C.; Hung, T.Y.; Lin, C.W.; Hsu, C.T. Video forgery detection using correlation of noise residue. Proc. of the IEEE Workshop on Multimedia Signal Processing. IEEE, 2008, pp. 170–174.

8. Farid, H. Image forgery detection. IEEE Signal Processing Magazine 2009, 26, 16–25.

9. Huaxiao Mo, B.C.; Luo, W. Fake Faces Identification via Convolutional Neural Network. Proc. of the ACM Workshop on Information Hiding and Multimedia Security. ACM, 2018, pp. 43–47.

10. Marra, F.; Gragnaniello, D.; Cozzolino, D.; Verdoliva, L. Detection of GAN-Generated Fake Images over Social Networks. Proc. of the IEEE Conference on Multimedia Information Processing and Retrieval, 2018, 274 pp. 384–389. doi:10.1109/MIPR.2018.00084.

11. Chollet, F. Xception: Deep learning with depthwise separable convolutions. Proc. of the IEEE conference on 276 Computer Vision and Pattern Recognition 2017, pp. 1610–02357.