

# Exposing the Mirage: Detecting Fake Faces amidst Real-world Variability

<sup>1</sup>VN. Divyasree, <sup>2</sup>Dr. S. Chinna Venkateshwarlu.

<sup>1</sup> Student from Department of Electronics and Communications Engineering, Institute of Aeronautical Engineering, Dundigal, Hyderabad, Telangana. Mail: [20951A0498@iare.ac.in](mailto:20951A0498@iare.ac.in)

<sup>2</sup> Professor from Department of Electronics and Communications Engineering, Institute of Aeronautical Engineering, Dundigal, Hyderabad, Telangana. Mail: [c.venkateshwarlu@iare.ac.in](mailto:c.venkateshwarlu@iare.ac.in)

## ABSTRACT

This paper discusses about the LBPNet, a machine learning Convolutional Neural Network (CNN) called LBPNET, designed to detect fake face images. The approach involves extracting Local Binary Patterns (LBP) from images and training a CNN model using LBP descriptor images. The trained model is then used to classify test images as either fake or non-fake. LBP is a widely used texture operator due to its discriminative power, computational simplicity, and robustness to grayscale changes. The LBP feature vector is created by comparing pixel values with their neighboring pixels, generating a binary representation, and computing a histogram. This feature vector can be processed using various machine learning algorithms for image classification tasks, such as face recognition or texture analysis.

**Keywords:** Forgery detection, GAN, Contrastive loss, deep learning, fully convolutional network, Convolutional Neural Networks (CNN), Image classification, Face recognition, Criminal identification

## 1. INTRODUCTION

"Video forgery detection using correlation of noise residue" addresses the critical issue of detecting video forgeries, which involve unauthorized alterations or manipulations of video content. In today's digital age, where video manipulation tools are readily available, ensuring the integrity and authenticity of video data has become increasingly important[1].

The paper focuses on the detection of image forgeries, which involve the manipulation or alteration of digital images with the intent to deceive viewers. Detecting such forgeries is crucial for maintaining the integrity and authenticity of visual information.[2]

This introduces a novel technique called progressive growing for training generative adversarial networks (GANs) to generate high-quality and diverse images. GANs are powerful models that consist of a generator network and a discriminator network, trained in a competitive manner to generate realistic images.[3]

It introduces the concept of generative adversarial networks (GANs), which are a class of deep

learning models used for generating realistic synthetic data, such as images, based on a training dataset. GANs consist of two main components: a generator network and a discriminator network, which are trained simultaneously in a competitive manner.[4]

Architectural guidelines for designing DCGAN models, including architectural choices for both the generator and discriminator networks. It emphasizes the importance of architectural stability and provides insights into the effects of different network configurations on the quality of the generated images.[5]

The Wasserstein GAN, a variant of GANs that addresses some of the limitations of traditional GAN frameworks. It presents a novel objective function based on Wasserstein distance and highlights the benefits of using the critic network to estimate the distance.[6]

It evaluated on various datasets, including CIFAR-10 and ImageNet, demonstrating significant improvements over the original WGAN formulation. The experiments show that the improved training of WGANs leads to higher-quality sample generation, improved convergence behavior, and increased stability during training.[7]

## 1.1 LITERATURE SURVEY

Several studies have investigated the use of neural information processing systems and deep convolutional networks.

- i. **C.C.Hsu, et al. (2017)** The author propose a novel approach that focuses on utilizing the correlation of noise residue in video frames for forgery detection. The noise residue is the discrepancy between the original video frame and its estimated noise-free version.[1]
- ii. **H. Farid (2018):** Author used a comprehensive survey of existing image forgery detection techniques, highlighting their strengths, limitations, and applicability to different types of forgeries. It discusses both active and passive approaches to forgery detection, which involve techniques such as watermarking, statistical analysis, and digital forensic analysis.[2]
- iii. **Karras, Tero, et al.(2018)** proposed a progressive growing approach where the generator and discriminator networks are gradually increased in size and complexity during the training process .[3]
- iv. **I. Goodfellow, et a (2019)** used a novel framework where the generator network generates synthetic data samples, while the discriminator network distinguishes between real and generated samples.[4]

- v. **A. Radford, et al. (2019).** The advancement of unsupervised representation learning using deep convolutional generative adversarial networks.[5]
- vi. **M. Arjovsky, et al. (2017).** The theoretical properties of WGANs, including the Lipschitz continuity constraint on the critic network and the impact on training stability.[6]
- vii. **Gulrajani, Ishaan, et al. (2020).** proposed techniques, such as the gradient penalty and one-sided label smoothing, address the challenges of enforcing Lipschitz constraint and gradient stability.[7]

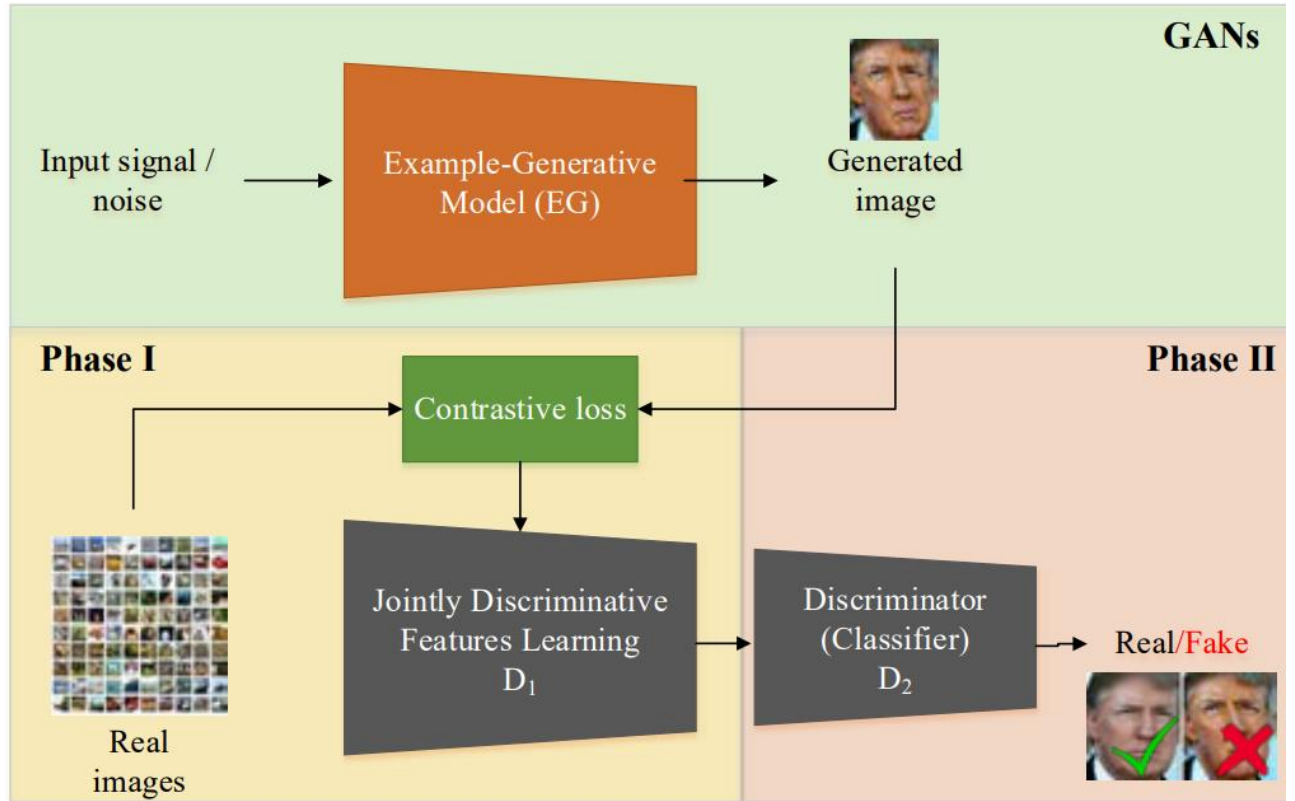
## 2. EXISTING METHODOLOGY

The flowchart of the proposed DeepFD. In the proposed method, we have two learning phases. First, we collect a lot of fake images synthesized by several GANs called example-generative model and real images to learn the jointly discriminative features D1 based on the proposed contrastive loss. Afterward, a discriminator (classifier) D2 will be concatenated to the D1 to further distinguish fake images. In the test phase, it is easy to distinguish the test image whether it is fake or real by D1 and D2 directly. The details of network architectures in D1 and D2 are respectively described in Table I. Note, the classifier D2 is directly concatenated to the 4th layer of the jointly discriminative feature learning network D1.

The jointly discriminative feature representation has learned, there are several ways to classify the received images such as SVM, random forest classifier, or Bayer classifier. In this work, we directly concatenate a convolutional layer and a fully connected layer to the network D1. In this way, the proposed DeepFD will be an end-to-end architecture.

The methodology for detecting fake face images in the wild involves collecting a diverse dataset of real and manipulated images, preprocessing the data, extracting features using deep learning models, training a classification model, and validating its performance. Optimization techniques are applied to improve the model, and once deployed, it can be integrated into real-world applications for real-time or large-scale detection of fake face images.

## 2.1 EXISTING BOLCK DIAGRAM



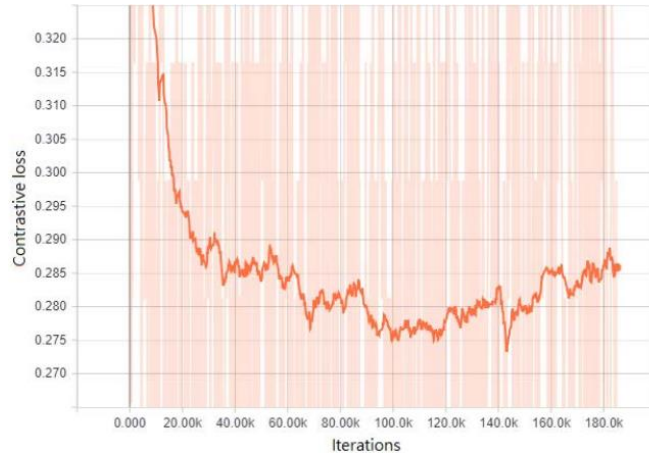
**Figure 2.1.1: Flowchart of the proposed deep forgery discriminator**

Network structures in jointly discriminator:

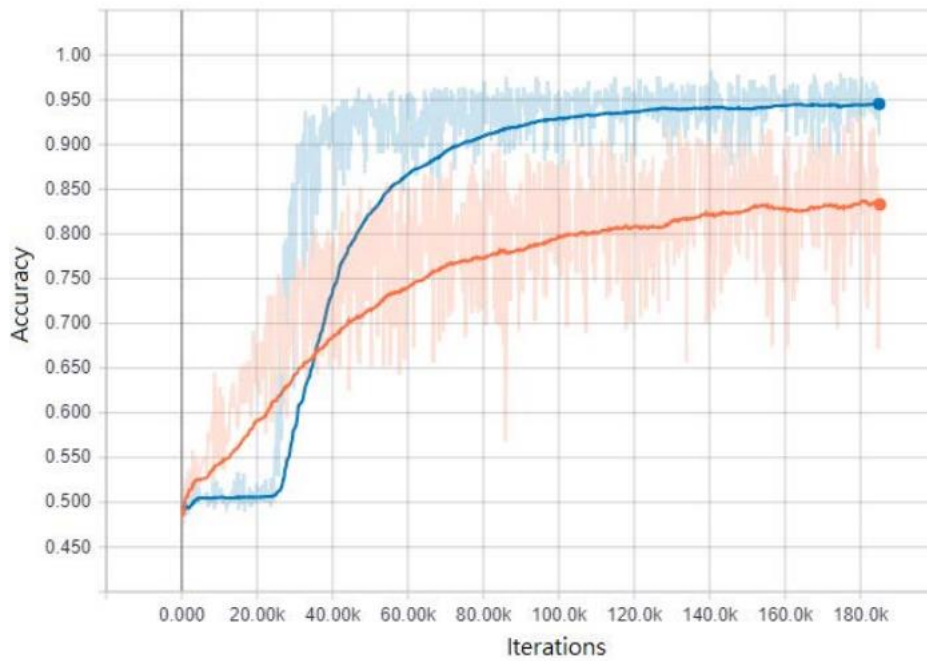
Layers	$D_1$	$D_2$
1	Conv. layer, kernel=7*7, stride=4, channel=96	Conv. layer, kernel=3*3, channel = 2
2	Residual block *2, channel=96	Global average pooling
3	Residual block *2, channel=128	Fully connected layer, neurons=2
4	Residual block *2, channel=256	Softmax layer
5	Fully connected layer, neurons=128 Softmax layer	

$$= \frac{1}{2} (p_{ij}(E_W)^2 + (1 - p_{ij})(\max(0, m - E_W))^2),$$

where  $E_W = \|D_1(\mathbf{x}x_1) - D_1(\mathbf{x}x_2)\|$  and  $m$  is the predefined marginal value. In this manner, it is possible to learn the common characteristic of the fake images generated by different GANs.



**Figure 2.1.2: The curve of the contrastive loss for learning D1 using pairwise information.**



**Figure 2.1.3: The performance comparison between the proposed DeepFD with (Blue line) / without (Orang line) contrastive loss for training set excluding LSGAN.**

Method	Exclusive of LSGAN		Exclusive of DCGAN		Exclusive of WGAN		Exclusive of WGAN-GP		Exclusive of PGGAN	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Traditional method [3]	0.205	0.580	0.253	0.774	0.235	0.673	0.242	0.604	0.222	0.862
CNN+RFC	0.547	0.566	0.563	0.585	0.546	0.603	0.580	0.536	0.520	0.531
CNN+SVM	0.614	0.630	0.561	0.593	0.570	0.589	0.602	0.563	0.547	0.570
CNN+LC	0.599	0.567	0.547	0.521	0.563	0.539	0.582	0.583	0.500	0.474
DeepFD w/o contrastive loss	0.836	0.801	0.760	0.721	0.724	0.722	0.706	0.687	0.759	0.766
The proposed DeepFD	<b>0.947</b>	<b>0.922</b>	<b>0.871</b>	<b>0.844</b>	<b>0.838</b>	<b>0.847</b>	<b>0.818</b>	<b>0.835</b>	<b>0.926</b>	<b>0.918</b>

**Table1: The performance characteristics between the proposed DeepFD and other methods.**

### 3. PROBLEM IDENTIFICATION

The project "Learning to Detect Fake Face Images in the Wild" aims to address the problem of identifying fake face images, it is important to acknowledge its potential limitations. Some of the common problems that have been identified in the literature include:

**Generalization to evolving techniques:** The project's effectiveness may be limited by its ability to generalize to new and evolving techniques for generating fake face images. As adversaries develop more sophisticated manipulation methods, the model may struggle to detect these emerging types of fakes, requiring continuous updates and adaptation.

**Lack of diverse training data:** The availability of a diverse and comprehensive dataset of real and fake face images is crucial for training an accurate model. Limited or biased training data may result in reduced performance or biased detection, particularly if certain types of fake face images are underrepresented.

**Adversarial attacks:** Adversaries may deliberately craft fake face images to evade detection by exploiting vulnerabilities in the model. Adversarial attacks can manipulate the model's decision-making process, leading to false positives or false negatives, thereby reducing the reliability of the system.

**Ethical and privacy concerns:** The project must consider the ethical and privacy implications of analyzing and detecting face images. Ensuring data privacy, consent, and fair use of facial data is crucial to avoid potential harms and maintain user trust.

**Real-time processing and computational requirements:** The detection of fake face images in real-world scenarios often requires real-time processing, which can be computationally intensive. The project may face limitations in terms of computational resources, latency, and scalability, impacting its applicability in real-time applications or systems with high throughput requirements.

We considered these limitations in the methodology for detecting fake face images, and to continuously evaluate and update the system to address emerging issues and improve its performance.

### 3.1 PROPOSED METHODOLOGY

Some of the common approaches that have been proposed in the literature include:

**Continuous model updates:** To address evolving manipulation techniques, the project should establish a framework for continuous model updates. Regularly monitoring emerging fake image generation methods and incorporating new training data can help the model stay up-to-date and improve its ability to detect new types of fakes.

**Diverse and representative training data:** Collecting a diverse and representative dataset is crucial. Efforts should be made to gather a wide range of real and manipulated face images, ensuring that various types of manipulations are adequately represented. Collaborations with research institutions and partnerships with organizations working in the field can help access a more comprehensive dataset.

**Adversarial robustness techniques:** Implementing adversarial robustness techniques can enhance the model's resilience against adversarial attacks. Techniques like adversarial training, input perturbations, or defensive distillation can be explored to improve the model's ability to detect manipulated images that attempt to evade detection.

**Ethical considerations and transparency:** The project should prioritize ethical considerations and transparency. Adhering to privacy regulations, obtaining appropriate consent, and ensuring responsible use of facial data are essential. Providing clear explanations of the detection process and being transparent about the model's limitations and potential biases can help build user trust and address privacy concerns.

**Optimization for real-time processing:** To enable real-time detection, optimizing the model and the overall system for efficient processing is necessary. This can involve techniques such as model compression, hardware acceleration, and algorithmic optimizations to reduce computational requirements and latency.

Thus, by applying these methods, it may be possible to overcome the challenges and limitations of and improve their accuracy and robustness in practice.

### 3.2 PROPOSED BLOCK DIAGRAM

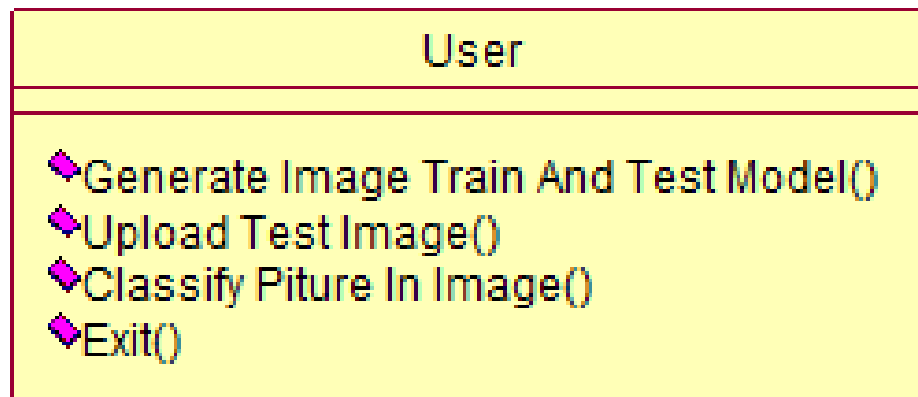


Figure 3.2.1: class diagram

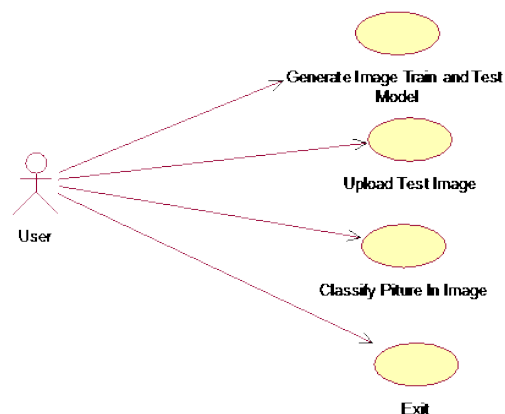
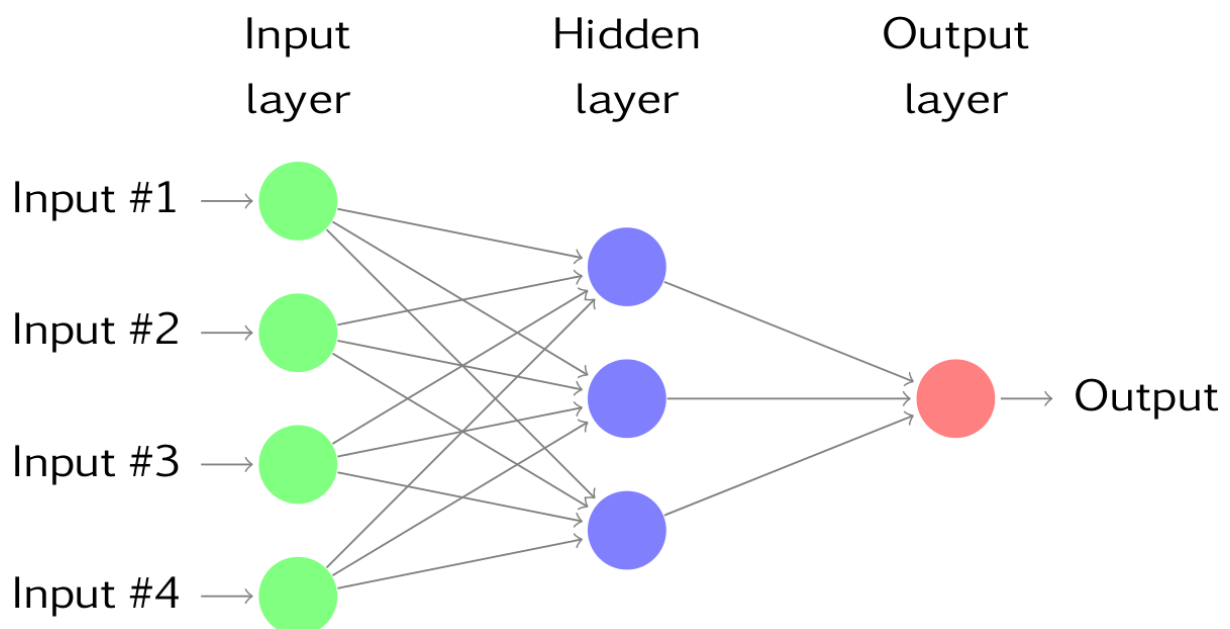
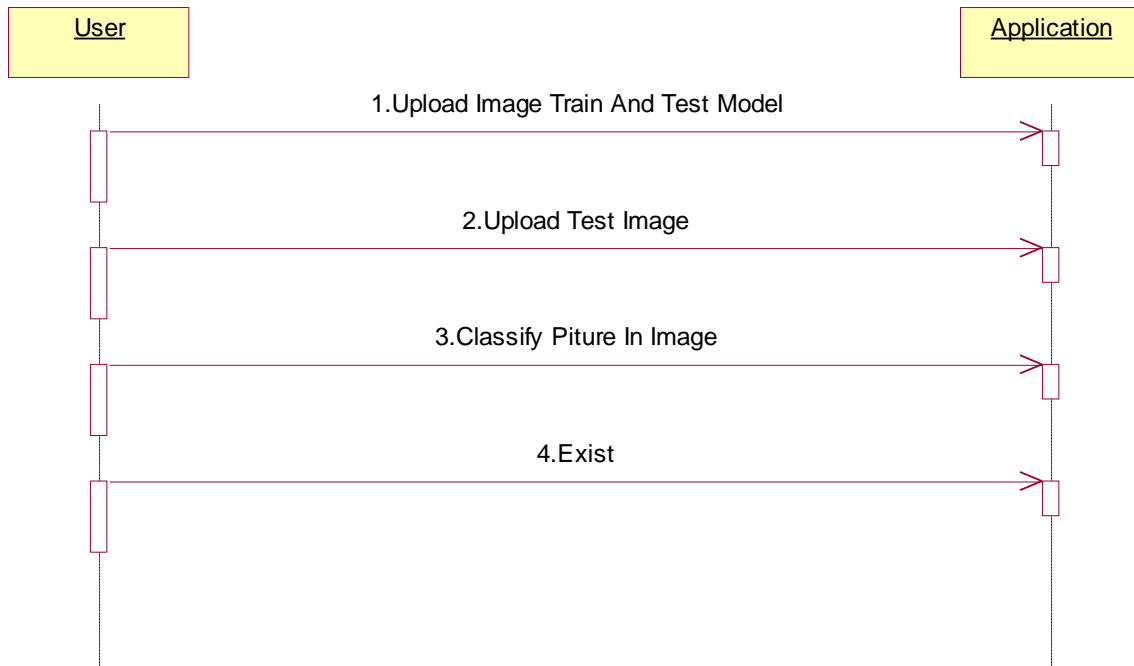


Figure 3.2.2: Usecase Diagram

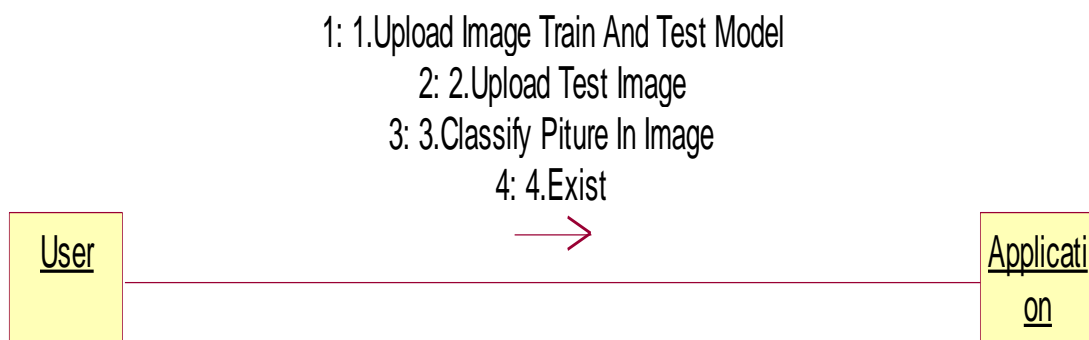




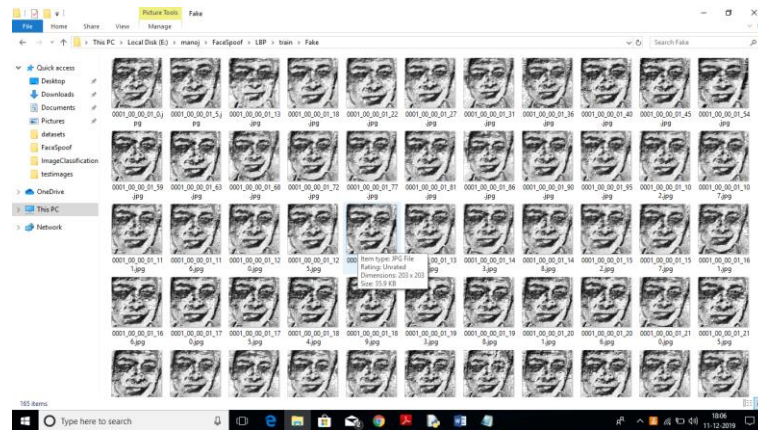
To predict image class multiple layers operate on each other to get best match layer and this process continues till no more improvement left.



**Figure 3.2.3: Sequence Diagram**



**Figure 3.2.4: collaboration diagram**



**Figure 3.2.5: All this fake and real images you can see inside ‘LBP/train’ folder.**

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) **Dataset Details:** In this paper author has used NUAA Photograph Imposter (fake) Database with images obtained from real and fake faces. We also used images and convert that image into LBP format. Below are some images from LBP folder.

(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.

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.

## 4. PRACTICAL SETUP

To set up a practical implementation of neural information processing systems and deep convolutional networks, here are some steps that may be helpful:

**Data collection and preprocessing:** Gather a diverse dataset of real and manipulated face images. This can involve sourcing publicly available datasets, collaborating with research institutions or organizations, and utilizing appropriate data collection techniques. Preprocess the collected data by resizing images, normalizing pixel values, and augmenting the dataset to increase its diversity.

**Model development:** Select a suitable deep learning architecture, such as a convolutional neural network (CNN), for feature extraction and classification. Implement the chosen model using a deep learning framework, such as TensorFlow or PyTorch. Train the model using the preprocessed dataset, optimizing the model parameters through techniques like gradient descent and backpropagation.

**Wavelet transform:** Apply wavelet transform to the preprocessed streamflow data to decompose it into different frequency components. Choose an appropriate wavelet basis function and decomposition level based on the characteristics of the data.

**Validation and evaluation:** Split the dataset into training and testing sets. Evaluate the trained model's performance using appropriate evaluation metrics, including accuracy, precision, recall, and F1 score. Conduct thorough testing to ensure the model's ability to accurately detect fake face images.

**Optimization and fine-tuning:** Analyze the model's performance and identify areas for improvement. Fine-tune the model by adjusting hyperparameters, modifying the architecture, or incorporating regularization techniques to enhance its accuracy and robustness. Continuously evaluate the model's performance and iterate on the optimization process as needed.

**Deployment and integration:** Once the model has been optimized, deploy it in a practical setup. This can involve integrating the model into an application, system, or platform where it can be utilized for real-time or large-scale detection of fake face images. Consider the hardware and software requirements, scalability, and efficiency to ensure smooth integration and performance.

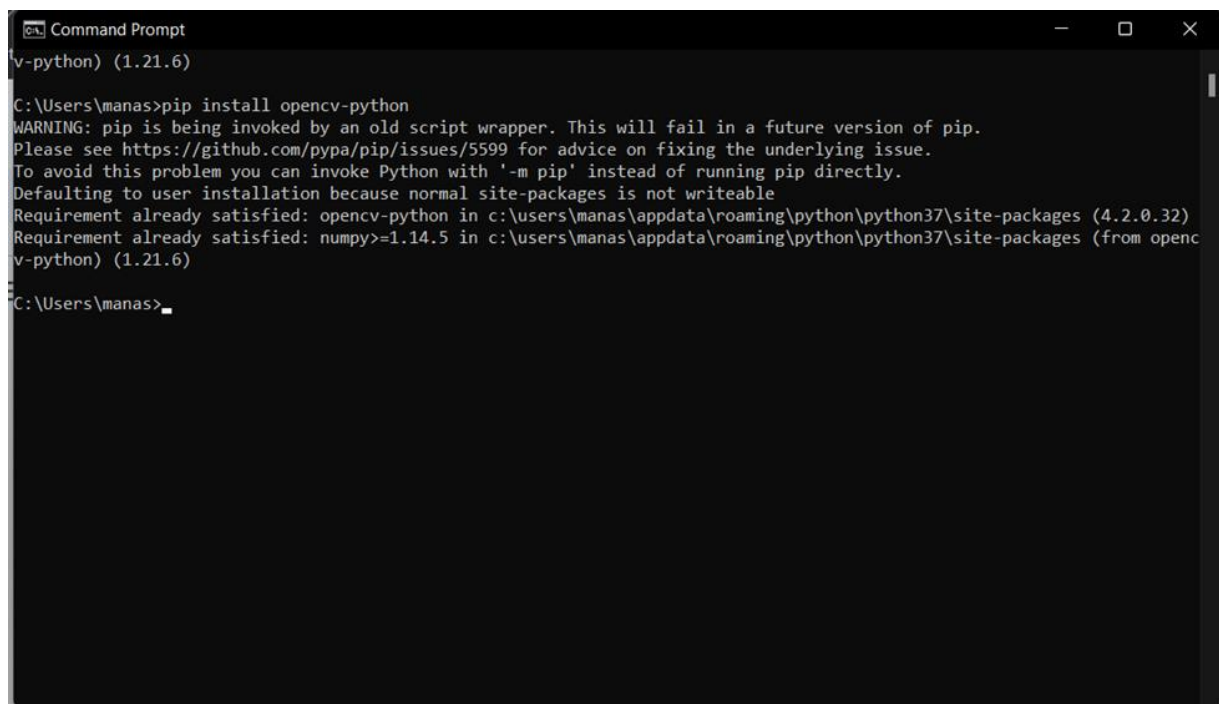
## 4.1 SOFTWARE USED

**Python:** Python is a versatile programming language that can be used for data analysis and machine learning. It has several libraries that can be used for wavelet transform analysis and neural network modeling, including Py Wavelets, SciPy, and TensorFlow.

**Jupyter:** Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It provides an interactive computing environment where you can write and execute code, visualize data, and present your analysis in a single document.

**Colab:** Google Colab is a cloud-based Jupyter Notebook environment provided by Google. It allows you to write and execute Python code directly in your web browser without the need for any local installation or setup. Colab provides free access to computing resources, including GPUs and TPUs, which can be beneficial for running machine learning models and data-intensive tasks.

## 5. INSTALLATION



```
Command Prompt
v-python) (1.21.6)

C:\Users\manas>pip install opencv-python
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: opencv-python in c:\users\manas\appdata\roaming\python\python37\site-packages (4.2.0.32)
Requirement already satisfied: numpy>=1.14.5 in c:\users\manas\appdata\roaming\python\python37\site-packages (from opencv-python) (1.21.6)

C:\Users\manas>
```

**Figure 4.1: Installation of open-cv using command prompt**

```
Command Prompt
(3.2.2)

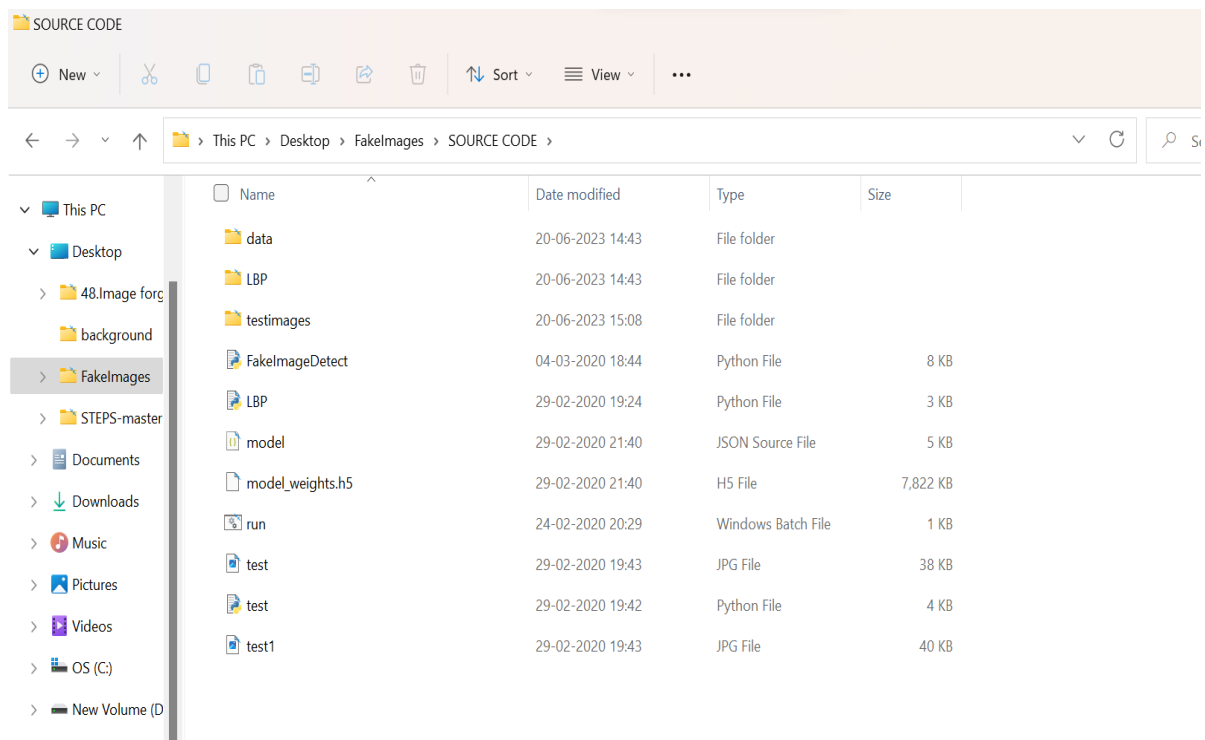
C:\Users\manas>python -m pip install --upgrade pip
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in c:\users\manas\appdata\roaming\python\python37\site-packages (23.1.2)

C:\Users\manas>pip install tensorflow
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow in c:\users\manas\appdata\roaming\python\python37\site-packages (2.11.0)
Requirement already satisfied: tensorflow-intel==2.11.0 in c:\users\manas\appdata\roaming\python\python37\site-packages
(from tensorflow) (2.11.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\manas\appdata\roaming\python\python37\site-packages (from tens
orflow-intel==2.11.0->tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\manas\appdata\roaming\python\python37\site-packages (from t
ensorflow-intel==2.11.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\manas\appdata\roaming\python\python37\site-packages (from te
nsorflow-intel==2.11.0->tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\manas\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\manas\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (0.2.0)
```

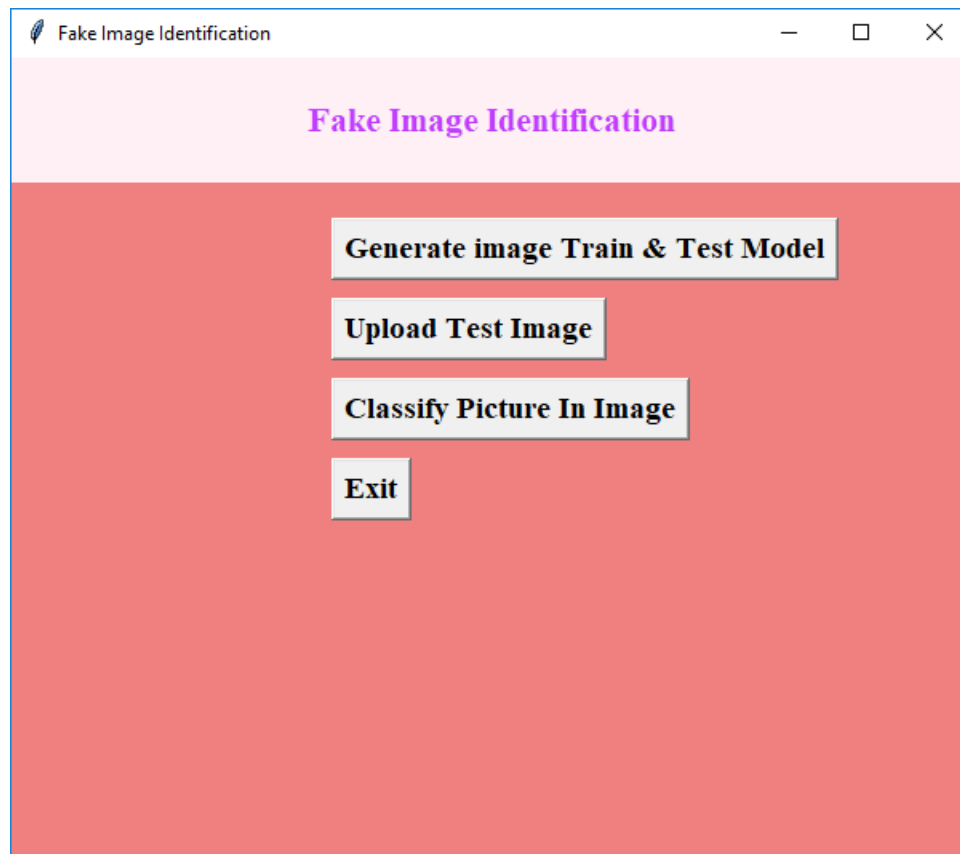
**Figure 4.2: Installation of TensorFlow using Command prompt**

## 6.RESULTS AND DISCUSSIONS

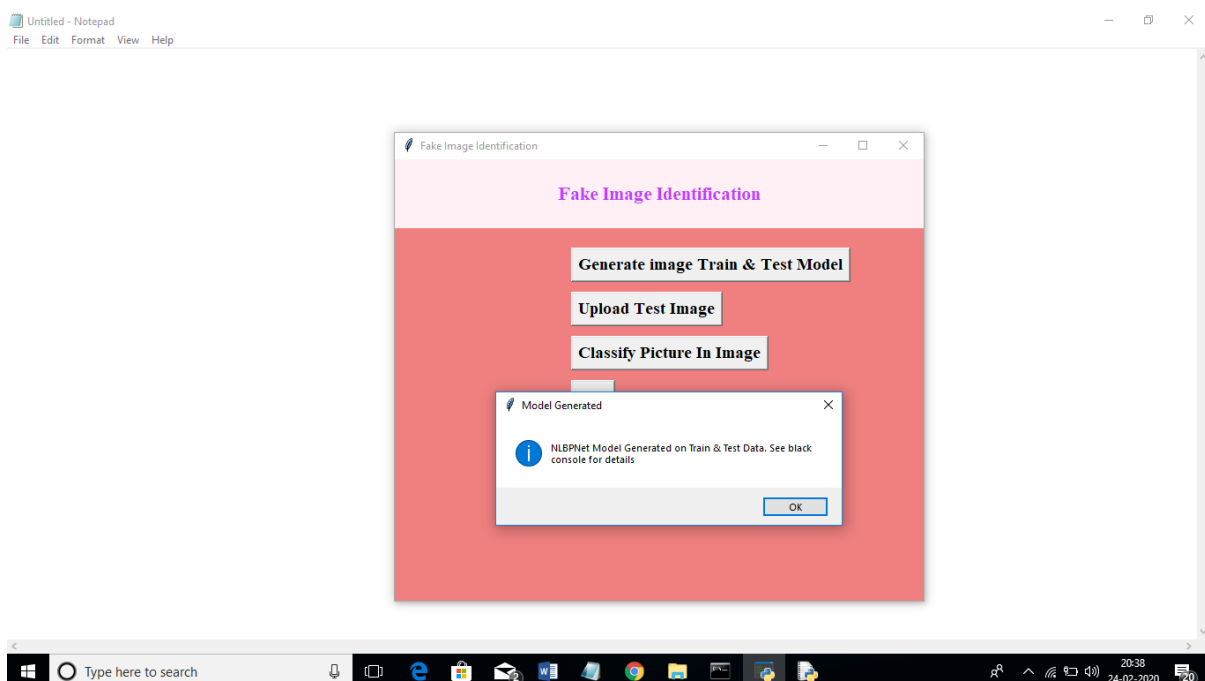
To run this project click on run.bat.



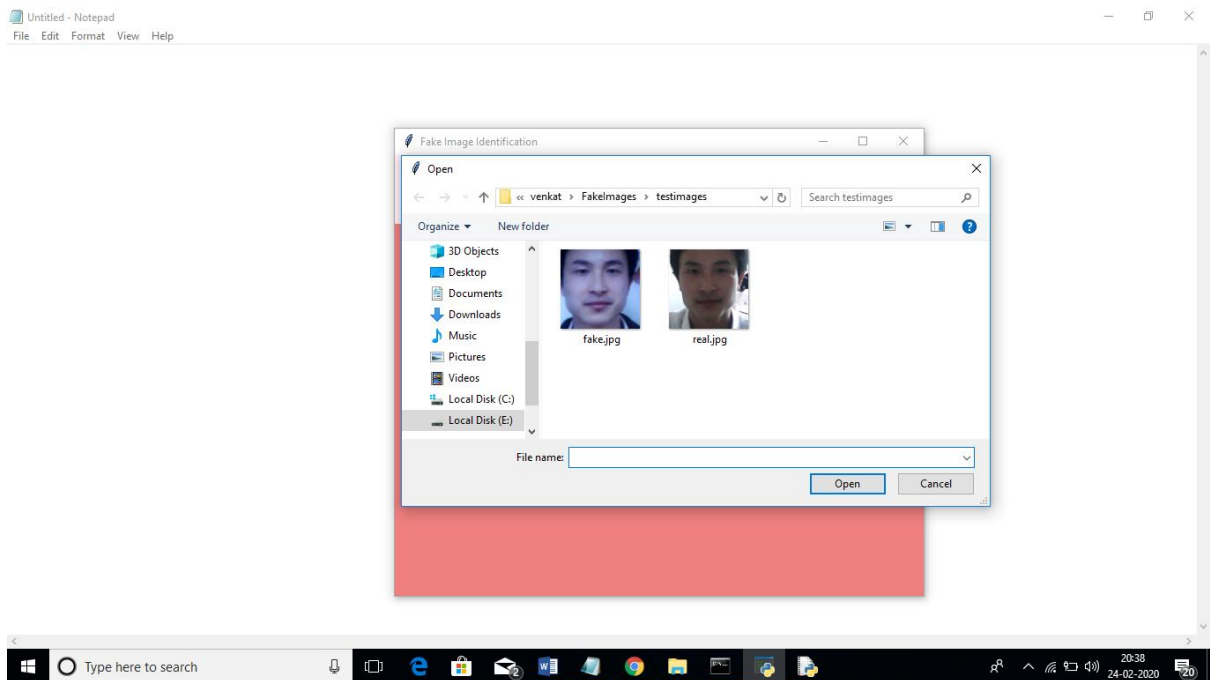
**Figure 6.0.1: Running the code in run.bat**



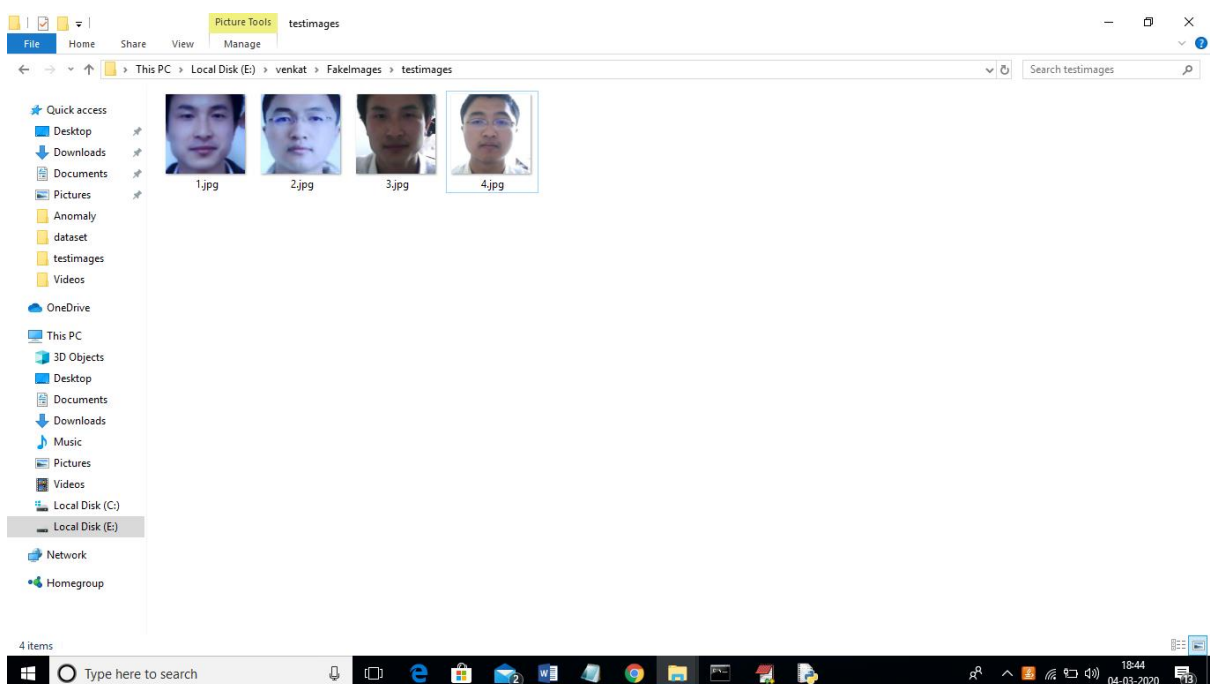
In above screen click on ‘Generate Image Train & Test Model’ button to generate CNN model using LBP images contains inside LBP folder.



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

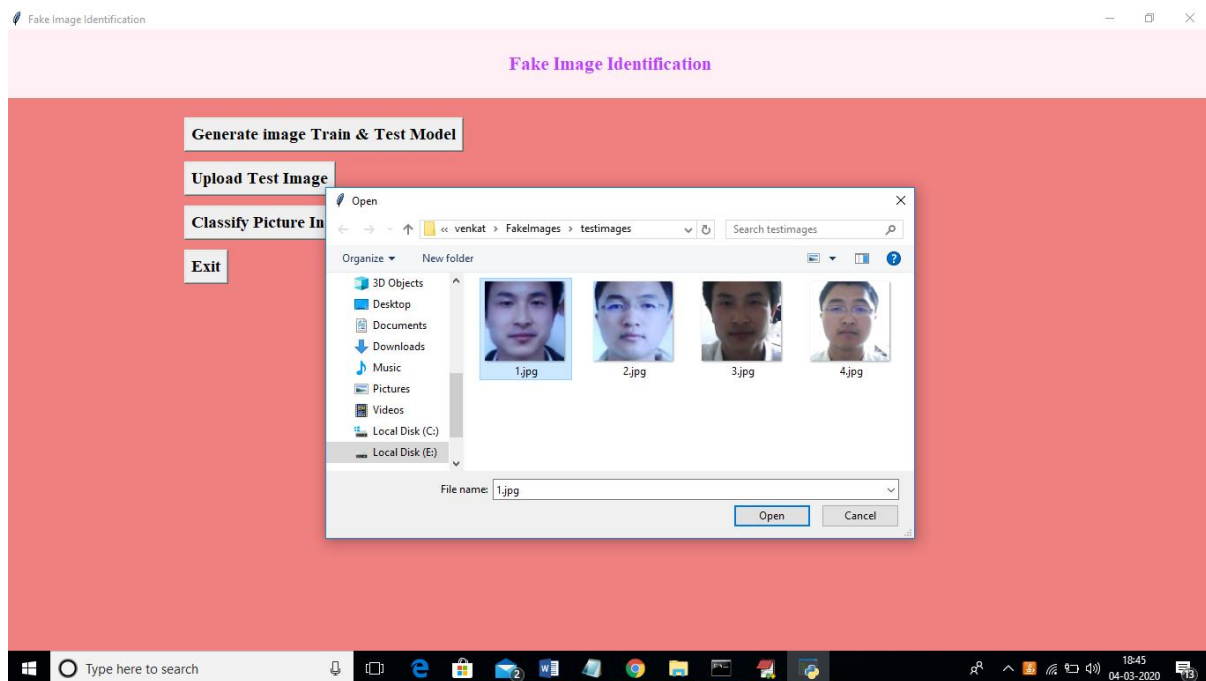


In above screen we can see two faces are there from same person but in different appearances. For simplicity I gave image name as fake and real to test whether application can detect it or not. In above screen I am uploading fake image and then click on ‘Classify Picture In Image’ button to get below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

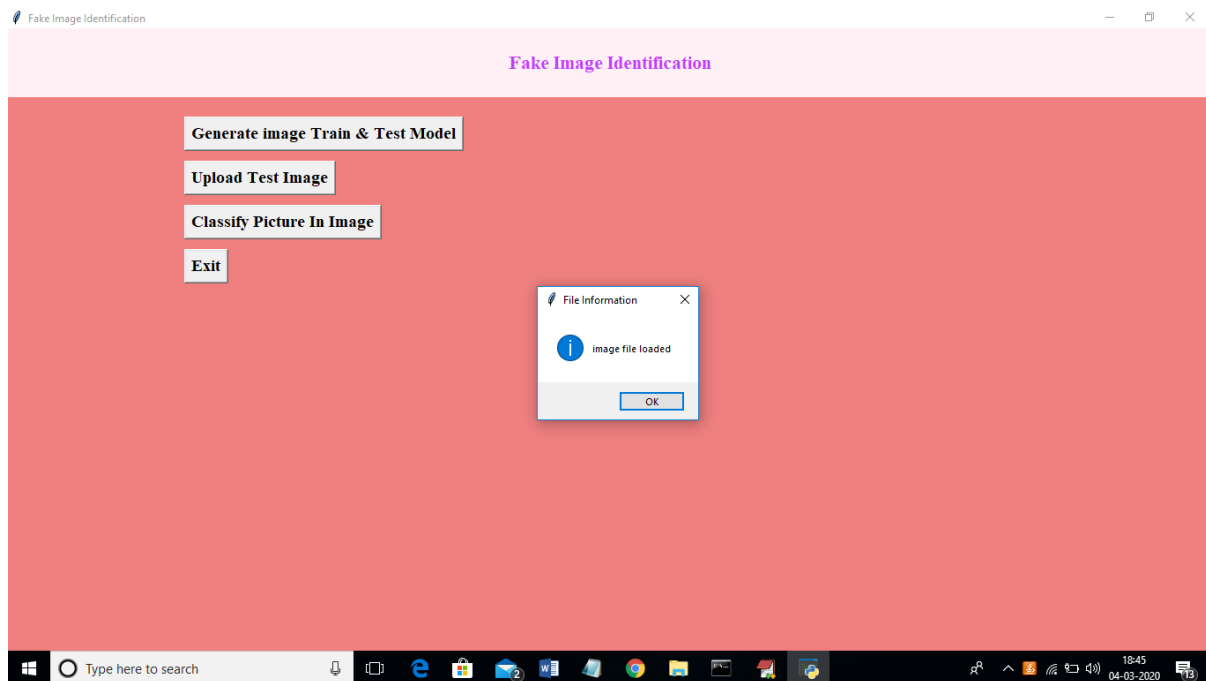


In above screen we can see all real face will have normal light and in fake faces peoples will try some

editing to avoid detection but this application will detect whether face is real or fake.

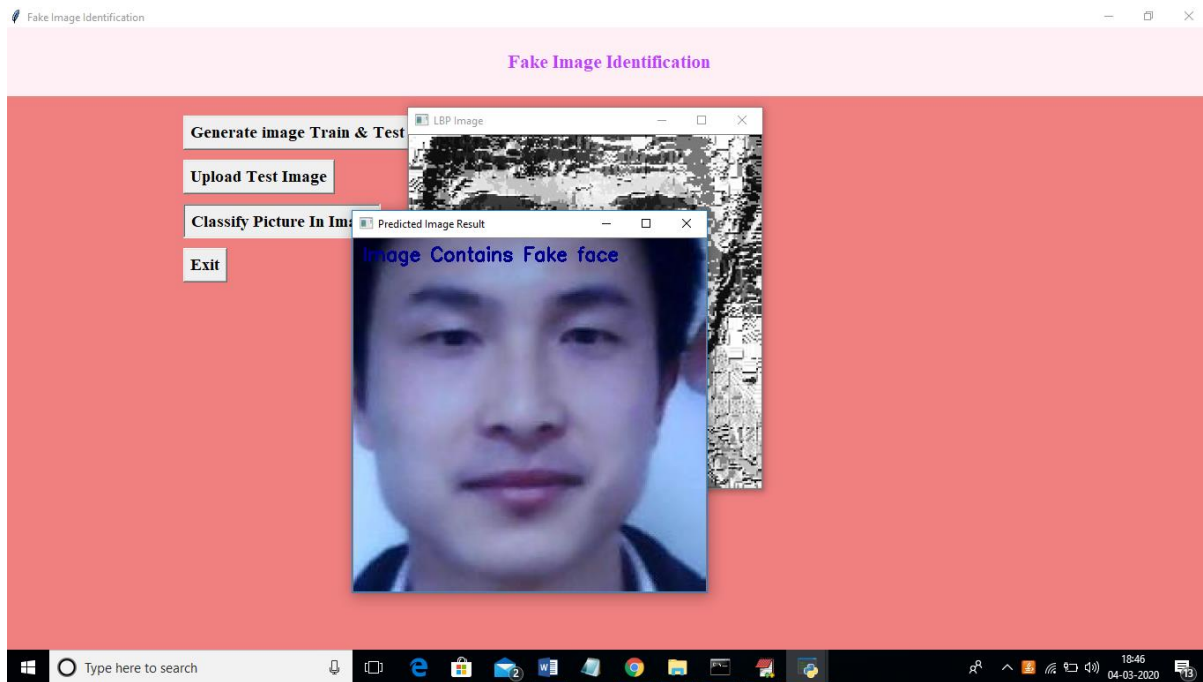


In above screen I am uploading 1.jpg and after upload click on open button to get below screen.



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





In above screen we are getting result as image contains Fake face. Similarly u can try other images also. If u want to try new images then u need to send those new images to us so we will make CNN model to familiar with new images so it can detect those images also.

## 7. CONCLUSION AND FUTURE SCOPE:

This project addresses the critical challenge of identifying manipulated or synthetic face images in real-world scenarios. By developing a methodology that combines data collection, preprocessing, feature extraction, model training, validation, and deployment, the project aims to accurately detect and distinguish between real and fake face images. Although the project has certain limitations, such as generalization to evolving techniques and ethical considerations, proposed solutions like continuous model updates, diverse training data, adversarial robustness techniques, ethical considerations, and real-time optimizations can help overcome these challenges.

Future scope for this project includes several avenues for further exploration and improvement. Firstly, ongoing research and development can focus on enhancing the model's ability to detect emerging manipulation techniques and novel types of fake face images. Continuous updates to the model's training data and architecture will be crucial in adapting to evolving threats.

Additionally, the project can extend its scope by considering multi-modal approaches that incorporate additional cues, such as audio or text, to further enhance the detection of manipulated content. By leveraging multiple modalities, the system can gain more robustness and accuracy in identifying fake face images.

## ACKNOWLEDGMENTS

I would like to extend my deepest appreciation to my research advisor, [Dr. S. China Venkateshwarlu], for their unwavering support and guidance. Their expertise, insightful feedback, and continuous encouragement have been instrumental in shaping the direction and quality of my research. Their dedication and commitment to my academic growth have been truly inspiring, and I am grateful for the opportunity to work under their mentorship.

## REFERENCES

1. **C.C.Hsu, et al. (2017)** The author propose a novel approach that focuses on utilizing the correlation of noise residue in video frames for forgery detection. The noise residue is the discrepancy between the original video frame and its estimated noise-free version.[1]
2. **H. Farid (2018):** Author used a comprehensive survey of existing image forgery detection techniques, highlighting their strengths, limitations, and applicability to different types of forgeries. It discusses both active and passive approaches to forgery detection, which involve techniques such as watermarking, statistical analysis, and digital forensic analysis.[2]
3. **Karras, Tero, et al.(2018)** proposed a progressive growing approach where the generator and discriminator networks anre gradually increased in size and complexity during the training process .[3]
4. **I. Goodfellow, et a (2019)** used a novel framework where the generator network generates synthetic data samples, while the discriminator network distinguishes between real and generated samples.[4]

5. **A. Radford, et al. (2019).** The advancement of unsupervised representation learning using deep convolutional generative adversarial networks.[5]
6. **M. Arjovsky, et al. (2017).** The theoretical properties of WGANs, including the Lipschitz continuity constraint on the critic network and the impact on training stability.[6]
7. **Gulrajani, Ishaan, et al. (2020).** proposed techniques, such as the gradient penalty and one-sided label smoothing, address the challenges of enforcing Lipschitz constraint and gradient stability.[7]

## APPENDIX

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os

def get_pixel(img, center, x, y):
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1
    except:
        pass
    return new_value

def lbp_calculated_pixel(img, x, y):
    center = img[x][y]
    val_ar = []
    val_ar.append(get_pixel(img, center, x-1, y+1))    # top_right
    val_ar.append(get_pixel(img, center, x, y+1))      # right
    val_ar.append(get_pixel(img, center, x+1, y+1))    # bottom_right
    val_ar.append(get_pixel(img, center, x+1, y))      # bottom
    val_ar.append(get_pixel(img, center, x+1, y-1))    # bottom_left
    val_ar.append(get_pixel(img, center, x, y-1))     # left
    val_ar.append(get_pixel(img, center, x-1, y-1))    # top_left
    val_ar.append(get_pixel(img, center, x-1, y))      # top
```

```

power_val = [1, 2, 4, 8, 16, 32, 64, 128]
val = 0
for i in range(len(val_ar)):
    val += val_ar[i] * power_val[i]
return val

```

```

def main():
    filename = 'data/Fake'
    for root, dirs, files in os.walk(filename):
        for fdata in files:
            image_file = root+"/"+fdata;
            img_bgr = cv2.imread(image_file)
            height, width, channel = img_bgr.shape
            img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
            img_lbp = np.zeros((height, width,3), np.uint8)
            for i in range(0, height):
                for j in range(0, width):
                    img_lbp[i, j] = lbp_calculated_pixel(img_gray, i, j)
            cv2.imwrite('LBP/validation/Fake/'+fdata, img_lbp)
            cv2.imwrite('LBP/train/Fake/'+fdata, img_lbp)

```

```

filename = 'data/Real'
for root, dirs, files in os.walk(filename):
    for fdata in files:
        image_file = root+"/"+fdata;
        img_bgr = cv2.imread(image_file)
        height, width, channel = img_bgr.shape
        img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
        img_lbp = np.zeros((height, width,3), np.uint8)
        for i in range(0, height):
            for j in range(0, width):
                img_lbp[i, j] = lbp_calculated_pixel(img_gray, i, j)
        cv2.imwrite('LBP/validation/Real/'+fdata, img_lbp)

```

```
cv2.imwrite('LBP/train/Real/'+fdata, img_lbp)  
print("LBP Program is finished")
```

```
if __name__ == '__main__':  
    main()
```











