A Project Report On

ENHANCING FACE RECOGNITION SECURITY: FACE ANTI-SPOOFING & LIVELINESS DETECTION USING HAAR CASCADE ALGORITHM

Mini project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY
(2021-2025)
BY

SRIRAM CHILIVERI	**21241A1256**
P. MAHESH BABU	**21241A1246**
YENAMALA VARSHIT	**21241A1265**

Under the Esteemed Guidance
of
Dr. R. V. S. S. S. NAGINI
Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
HYDERABAD

2023-24

CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled **"ENHANCING FACE RECOGNITION SECURITY: FACE ANTI-SPOOFING & LIVELINESS DETECTION USING HAAR CASCADE ALGORITHM"** done by **SRIRAM CHILIVERI (21241A1256), P. MAHESH BABU (21241A1246), YENAMALA VARSHIT (21241A1265)** of **B. Tech** in the Department of Information of Technology, **Gokaraju Rangaraju Institute of Engineering and Technology** during the period 2020-2024 in the partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

Dr. R. V. S. S. S. Nagini
Associate Professor
(Internal Guide)

Dr. Y. J. Nagendra Kumar
Head of the Department

(Project External)

ACKNOWLEDGEMENT

We take the immense pleasure in expressing gratitude to our Internal guide**,** **Mrs. R. V. S. S. S. Nagini, Associate Professor, Dept of IT**, GRIET. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. Y J Nagendra Kumar,** HOD IT, our Project Coordinators **Dr. R. V. S. S. S. Nagini** and **Mr. P.K. Abhilash** for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

Email: sriramchiliveri1544@gmail.com
Contact No: 8074294033

Email: mahesh9949644@gmail.com
Contact No: 9177795648

Email: varshithyenamala@gmail.com
Contact No. 7995140990

DECLARATION

This is to certify that the mini-project entitled "**ENHANCING FACE RECOGNITION SECURITY: FACE ANTI-SPOOFING & LIVELINESS DETECTION USING HAAR CASCADE ALGORITHM"** is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

SRIRAM CHILIVERI 21241A1256

P. MAHESH BABU 21241A1246

YENAMALA VARSHIT 21241A1265

TABLE OF CONTENTS

11 LIST OF DIAGRAMS

ABSTRACT

Face recognition has become an essential component in various security applications, such as access control, surveillance, and identity verification. However, traditional face recognition systems can be vulnerable to spoofing attacks, where an attacker tries to gain unauthorized access by presenting a fake face. To address this issue, we propose a novel approach that combines the Haar Cascade algorithm and Convolutional Neural Networks (CNN) to enhance the security of face recognition systems.

The Haar Cascade algorithm is a widely used technique for face detection, which uses a set of Haar-like features and an AdaBoost classifier to efficiently identify the presence of a face in an image. However, the Haar Cascade algorithm can be susceptible to variations in lighting, pose, and occlusions. To overcome these limitations, we integrate a Convolutional Neural Network (CNN) into the face recognition pipeline. The CNN architecture enables the system to learn robust facial features and better discriminate between real and fake faces, even in challenging environmental conditions. The results demonstrate significant improvements in detection accuracy and speed, reducing vulnerability to common attacks. This paper concludes that the Haar Cascade algorithm is a promising tool for developing resilient face recognition systems, with future work focused on enhancing its robustness in varied and challenging environments.

Keywords: Object Detection, Face recognition, CNN, Haar cascade Algorithm

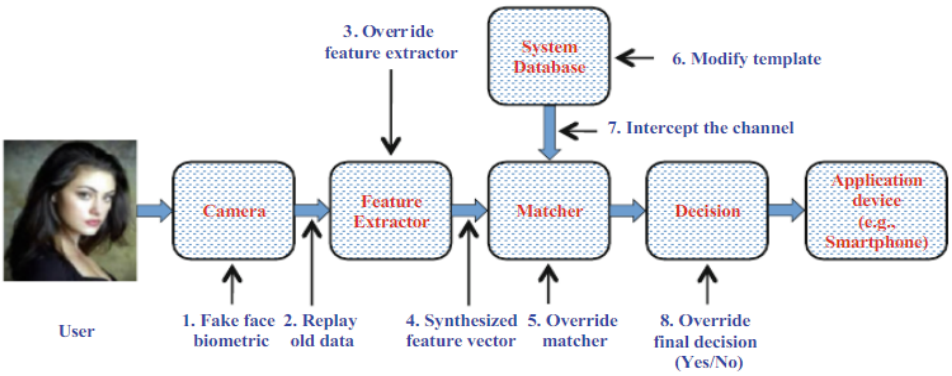
Domain: Machine learning

1. INTRODUCTION

1.1 Introduction to Project

In the last decade, many applications have integrated automatic face recognition systems because of their high richness of face cues that function as powerful biometrics for people's identification. Indeed, facial recognition is already present at the current stage of its development as a system. For instance, UIDAI confirms an identity to any individual in India who uses his or her face; on the other hand, Microsoft Kinect recognizes the face and the dashboard to log in the individual to the Xbox Live profile. In the same regard, another secured way of accessing control that has now become a common feature and an option to the traditional password is the facial biometrics that are currently integrated into the Android KitKat, the Lenovo VeriFace, Asus Smart-Logon and Toshiba SmartFace mobile operating systems.

Contrary to vectors such as reconciliation, the use of facial recognition systems also rises every year and their application in daily life is not unfamiliar. Due to this, the security loophole of facial recognition is gradually gaining awareness to the general public. However, the weaknesses of the facial systems under attack are largely ignored, whereas now it is possible to find Websites or simple educational videos to tell how one should hack facial systems to gain unauthorized access. It is seen from the figure, there are exactly eight points where security of the face recognition system can be tricked.



[Fig - 1]

FACE SPOOFING METHODS:

The term "face spoofing" describes the employment of deceitful strategies to lie to facial popularity software program into granting unapproved or fraudulent access. The ability of face spoofing will

become a chief hassle as facial popularity generation will become extra broadly used in lots of applications, together with cell devices, protection systems, and authentication procedures. Face spoofing is carried out the use of plenty of strategies, which may be divided into 4 categories: physical attacks, morphing attacks, multimodal attacks, and deepfake generation.

- **PHYSICAL ATTACKS**:
  - **Print Attack**: In a print attack, an attacker makes use of a high-resolution photo of the valid user`s face to impersonate and benefit unauthorized access. This technique is straightforward and regularly effective, in particular if the facial reputation machine lacks anti-spoofing measures.
  - **Replay Attack**: During a recording session, the attacker records a video or takes pictures of the person's face while they are authenticating properly. They then play back this recorded information to the facial recognition system when they try to log in again, deceiving the system into thinking they are the real user and allowing them access.
  - **3D Mask Attack**: This process includes fabricating a three-dimensional physical mask that closely resembles the genuine user's face. Sophisticated 3D printing methods enable offenders to create masks that are indistinguishable from real faces, making it possible to trick others into believing it's the actual person.

- **MORPHING ATTACKS**:

  Morphing attacks involve combining multiple facial features to create a composite image. For facial recognition algorithms, it is difficult to correctly recognize a person, because the resulting image combines the features of both people. This technique allows attackers to create a false identity that can be caught by authentication checks.

- **MULTIMODAL ATTACKS:**
  - **Mask Attack:** In a mask attack, the attacker employs a face mask as physical medium that mimics the original one (mask of the authorized user). It can be used in conjunction with another approach like the use of voice to enhance the attack.
  - **Texture & Pattern Attack:** A set pattern on the material or other minute patterns on the material can be utilized by the attackers. A Head probe or some other device can be used to distract the facial recognition system. These textures can confuse algorithms

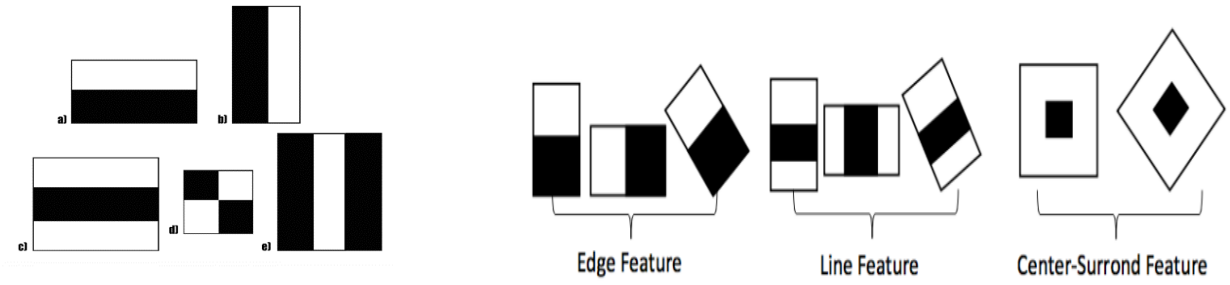
by incorporating some aspects that are most likely to be natural to the face of a human being.

- **DEEPFAKE GENERATION (GAN-BASED SPOOFING)**

  Generative adversarial networks or GAN for short, very powerful techniques for creating realistic synthetic content (such as deep fake movies) for facial speech. Generating artificial knowledge and capability of separating between authentic and fake information are the two major components of a GAN. The facial recognition algorithms get confused with between the concepts of the so-called fake and the underlying genuine 'identities' in facial speech since GANs can be trained to chime out fairly realistic facial images or movies.

HAAR CASCADE:

Haar Cascade is about direction detection of objects on images through the usage of features of an object. The cascade function is trained to use a set of multiple positive and negative images to get the detection. The complexity is not high and the given algorithm can be executed online in real time. We can train our cascade function for the common objects such as animals, cars, bike etc. Haar Cascade cannot be used for face recognition since it only detects the corresponding shape and size. Haar cascade employs the use of the cascade function together with the cascade window. It attempts to compile a list of features for each window and then distinguish between the positive and negative attributes of the situation. If window were to be an element in the object good and if not bad. Haar cascade can be described insofar as it is a binary classifier. This sets the positive for any staggered windows that may be a component of our object. and negative for the windows which are not allowed to be a part of our object.



[Fig - 2]

• Haar cascades can also operate in real time

• Haar cascade is not much accurate in comparison to the innovative object detection algorithms.

• It provides many false alarms. This can be optimized to some extent but they cannot be eliminated completely.

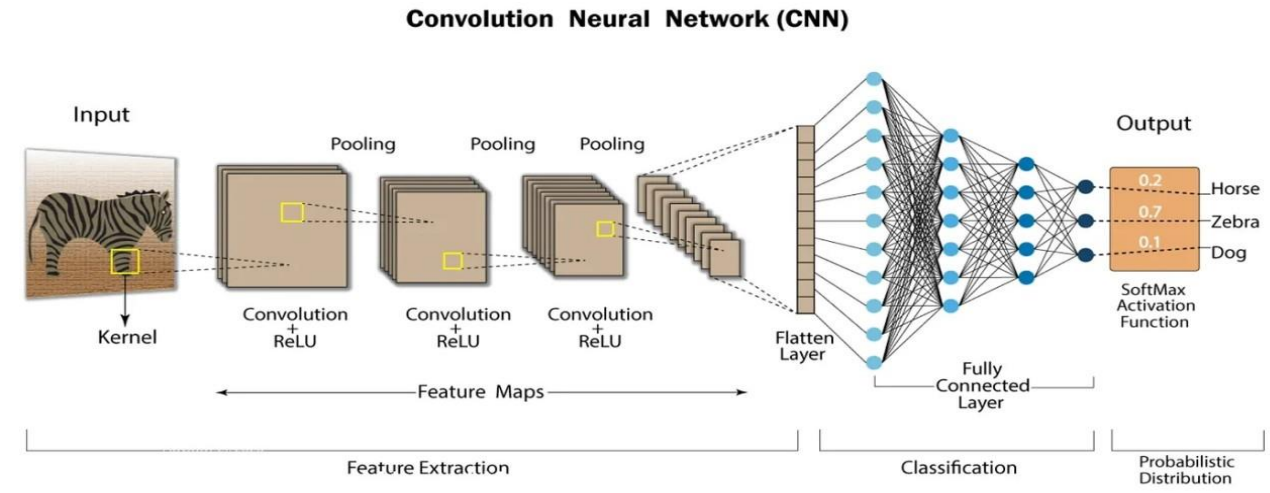
• It is very fit for implementation because of its simplicity.

• However, a major disadvantage of Haar Cascade is the detection of false negative.

CNN:

Modern face anti-spoofing systems rely heavily on Convolutional Neural Networks (CNNs) because of their remarkable capacity to extract intricate patterns and features from images. CNNs improve the detection of spoofing attempts in facial recognition systems when they are combined with the Haar Cascade algorithm, which is commonly used for object detection.

Through the analysis of grayscale image pixel intensity distribution, Haar Cascade classifiers are highly effective in identifying particular patterns. They work especially well for distinguishing between frontal faces and distinct facial features. In the context of anti-spoofing, faces in an image or video frame can be quickly located using Haar Cascade algorithms. By processing the regions of interest found by Haar Cascade, CNNs enhance this. They take out minute details that set authentic faces apart from spoofing materials like masks or printed pictures.



[Fig-3] CNN

CNNs improve the precision and resilience of anti-spoofing systems by automatically extracting discriminative features from training data through the use of deep learning techniques. Face anti-spoofing systems effectively mitigate various spoofing attacks by combining the feature learning capabilities of CNNs with the speed and accuracy of Haar Cascade for face detection. This ensures dependable authentication and security in facial recognition applications.

1.2 Existing System

The existing models mainly contain detection of real / spoof faces using Texture-Based Methods, Motion-Based Methods, Depth-Based Methods and Hardware-Based Solutions which intern use 3D cameras, Infrared sensors, R-CNN, YOLOv3 and YOLOv5.

- **Texture-Based**:
    - **Pros**: Simple and low computational cost.
    - **Cons**: Can be fooled by high-quality prints and screens.
- **Motion-Based**:
    - **Pros**: Effective against static image attacks.
    - **Cons**: Requires cooperation from the user.
- **Depth-Based**:
    - **Pros**: Strong against 2D spoofing attacks.
    - **Cons**: Requires specialized hardware..
- **Hardware-Based**:
    - **Pros**: Highly accurate and difficult to bypass.
    - **Cons**: Expensive and not always practical for all applications.

1.3 Proposed System

The proposed approach first employs the Haar Cascade algorithm to quickly identify potential face regions in an input image or video frame. These regions are then passed through the CNN-based face detection model, which performs a more detailed analysis to verify the authenticity of the detected faces.

2. REQUIREMENT ENGINEERING

2.1 Hardware Requirements

- Processor – i5 and above (64-bit OS).
- Memory – 4GB RAM (8GB or above is recommended for better performance)
- Input devices – Keyboard, Mouse, Webcam

2.2 Software Requirements

- Anaconda / Miniconda
- Python
- Python Libraries
  1.pandas
  2.numpy
  3.matplotlib
  4.tensorflow
  5.openCV

3. LITERATURE SURVEY

- Currently, Singh, Joshi and Nandi in proposed a face recognition system with live face detection by using eye and mouth movement. First, the system begins by conducting a liveness check for the user and in case the liveness check is complete, face recognition is performed. While the result indicates that although the actual system was-reported to achieve a good accuracy for a number of samples of different form of attacks that were created by the author, the liveness test was compromised by the cut-photo attacks where the eye and mouth region were removed completely.[1]

- Pupil tracking was adopted in the method developed by Killioğlu, Taşkiran and Kahraman. Their approach was to devise a way of observing the direction of the pupil using simple hardware components (the square frame that has a total compromise of eight Light Emitting Diodes; one for each direction). Although real subject tests reveal that the algorithm has a high success ratio, the approach used for real subjects has low robustness and generality. It can also be easily circumvented by a 3D mask or cut attack (to imitate blinking and moving the eyes). In addition, the latter will not be easy as it entails integrating it on to existing face recognition systems because of the extra hardware that is needed.[2]

- Dhawanpatil and Joglekar proposes to use Moiré patterns and Image Distortion Analysis (IDA) for anti spoofing. The authors divided the process into 3 modules:
    1. Subordinate module that receives the face of an individual that may be a genuine or a spoof face.
    2. In spoof detection here the spoof detection module, the face is checked for the existent of Moiré patterns and IDA if the face image is spoofed or not.
    3. The face matcher module were the image is classified as spoof or not with the help of an SVM classifier. [3]

- Anand and Gupta employed a spoof detection method that is a hybrid of SIFT descriptor, genetic algorithm and artificial neuron network in order to establish new features sets for the databases images into its respective category. Besides, the image-manipulation model of their experiment was assessed with a face spoof database of ". JPEG" images created by the authors

using the cameras with non-compression method. Thus, according to the result presented in the following table, the model has got quite high accuracy with an average of 97%. 86%, and the error rate 2 % respectively. 13% from the system, it was lacking in correctly recognizing real faces as per the performance figure that exhibits a rather high FRR of 53. 29%. [4]

- For the depth information Albakri and Alghowinem constructed the system that uses the 3D sensor of the Microsoft Kinect device. The identification and detection of faces done with the help of Haar-Cascade functions of OpenCV that corresponds to Kinect Sensor. For assessing the spoof test cases followed by the system, five different spoof test cases were examined as given below. This followed the observation that the size of the image & recursive transformations such as glasses, scarf & makeup did not affect the system as it was capable of detecting spoof images and get the real users in despite of the alteration. On the other hand, although the Kinect sensor's detectable distance range is 0. 7 - 6 m, the observation was made for the fact that whenever the test cases showed a distance of more than 1 meter, there is a drop in spoof detection accuracy. [5]

- To address the aforementioned issue, Thippeswammy, Vinutha, and Dhanapal came up with an approach in which they adopted multitable texture descriptors of the local appearance-based techniques. Their algorithm was then applied on the NUAA Photo Imposter database.
  From the presented results, there is an excellent display of performance in the detection of spoofed faces as 98. A very satisfactory accuracy was achieved in the identification of spoofed faces: only 39% of the time the identification was correct. On the other hand, uses the system in recognizing real faces was very low as only 51% of the real faces are recognized. [6]

- The proposed scheme for face anti-spoofing and liveness detection utilizes a CNN that processes aligned face images in RGB and HSV color spaces. The architecture includes multiple convolutional and pooling layers, incorporating skip connections inspired by ResNet, leading to a final soft-max layer for classification. The system ensures accurate alignment and cropping of faces to enhance feature detection. The cross-entropy function measures the loss, emphasizing spatial-structure information to distinguish between live and spoof faces.[7]

- The process involves acquiring 3D RGB training images, converting them to grayscale, reducing noise with a median filter, extracting the region of interest using the Prewitt method, enhancing contrast, and extracting features with LBP. A CNN classifier then classifies faces as real or fake, with performance compared to other methods.[8].

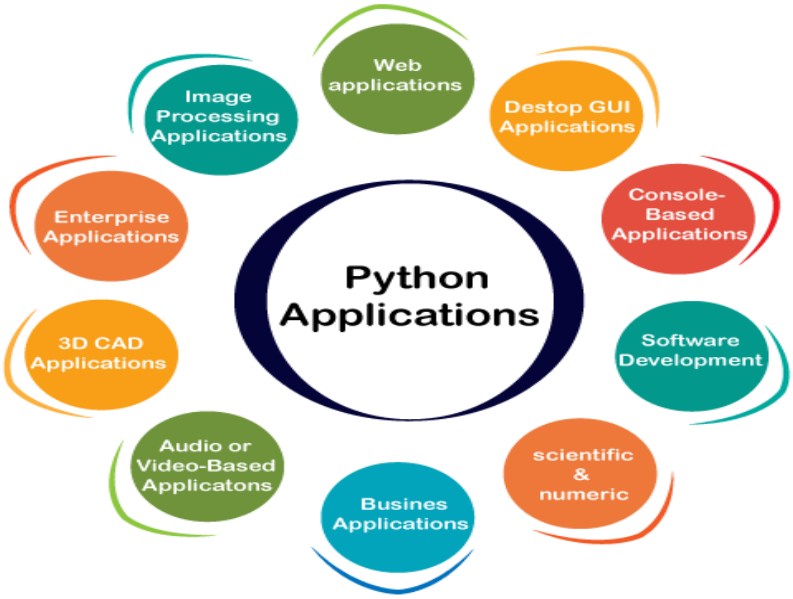
4. TECHNOLOGY

4.1 ABOUT PYTHON

The statistical analysis capabilities of Python are growing with the evolution of its environment. It successfully reconciles scale and class. Python places a high value on readability and performance. Its layout prioritizes program readability, and its basic syntax is easy enough for beginners to understand while still allowing programmers to express their code in fewer lines—especially when employing indentation. The dynamic system and automatic memory management system functions are this excessive degree language's specialty.

4.2 APPLICATIONS OF PYTHON

Python is used in many application domains. It makes its presence in every emerging field. It is the fastest-growing programming language and may be used to create any type of application.



[Fig – 4] Applications of python

4.3 PYTHON IS WIDELY USED IN DATA SCIENCE

We utilize Python, an open-source, flexible language, for data science. It provides functions related to mathematics and scientific functions. We presumably use it because of its extensive library and

straightforward syntax. Additionally, coding takes less time.
These are the main Python libraries used in data science.

4.3.1 PANDAS

A Python library for examining, cleaning, altering, and evaluating statistics. A dataset often includes a large number of useful and pointless statistics. They become more readable and pertinent with pandas.

4.3.2 NUMPY

A library in Python for reading, cleaning, investigating, and modifying statistics. A common dataset contains both valuable and useless data. They become readable and relevant with pandas.

4.3.3 MATPLOTLIB

A library is used for plotting graphs in python. It built on NumPy arrays. We can plot any graph from basic plot types to bar graph, histogram, scatter and many more.

4.3.5 CSV

it is a kind of file that stores tabular records, like a spreadsheet or a database. There are one or extra fields in each entry, that are separated by commas. We use the csv built in module to work with CSV files.

4.3 OPENCV

OpenCV (Open Source Computer Vision) is a open-source software library for computer vision and machine learning. It includes tools for basic tasks such as filtering, image transformations, and

morphological manipulation. It provides motion detection, object tracking, and background subtraction capabilities. OpenCV can record and manipulate video streams from cameras or files.

4.3.4 TENSORFLOW

Google created the open-source TensorFlow machine learning framework. TensorFlow provides a single API to distribute computation to one or more CPUs or GPUs in a desktop, server, or mobile device. Many tasks, including object identification, video analysis, and image classification, make extensive use of TensorFlow. To create, train, and implement an efficient face anti-spoofing model, TensorFlow offers an extensive collection of tools and packages.

4.4 DATASET DESCRIPTION

A dataset has been taken from the Kaggle in order to train our model. It consists of 4000 unique images. These images are in the json format. These images are extracted from diverse groups in kaggle website.

Input size: 153.8MB

----------Training Datasets----------

There are 2102 images in real directory

There are 2118 images in spoof directory

There are 4220 total images in training directory

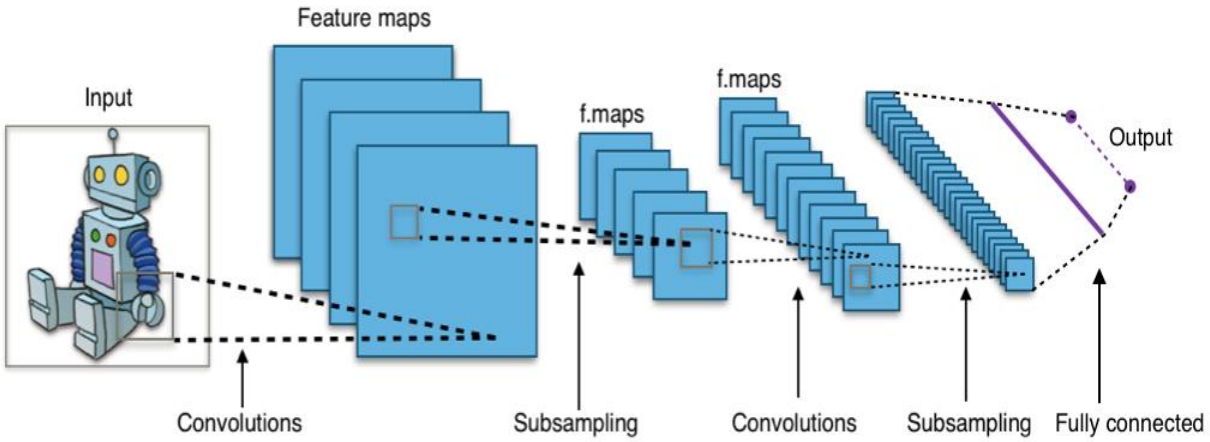
----------Testing Datasets----------

There are 477 images in real directory

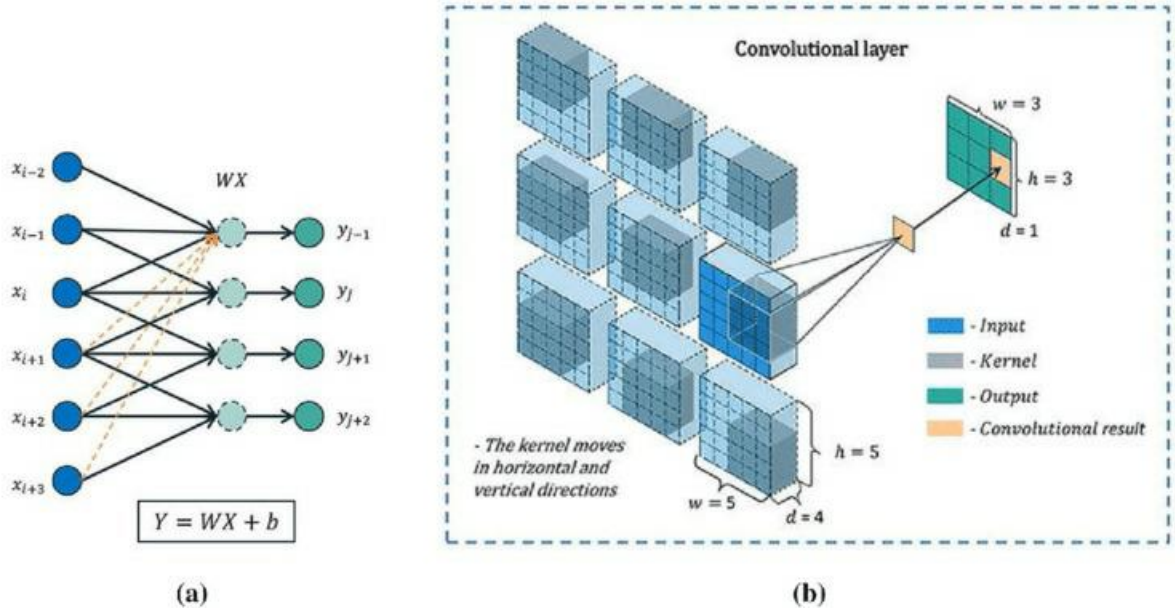
There are 474 images in spoof directory

There are 951 total images in testing directory

4.5 CNN



[Fig – 5] Layers of CNN



[Fig – 6]

5.DESIGN REQUIREMENT ENGINEERING

CONCEPT OF UML:

Unified Modeling Language (UML) is a well-defined language used to describe software systems and their components. It is a tool that offers a series of diagrams and shapes or symbols that shows the aspects of a system, parts of it as well as the relations of the various components; used in designing, describing and sharing the structure and the docket of any software development project among the various people involved.

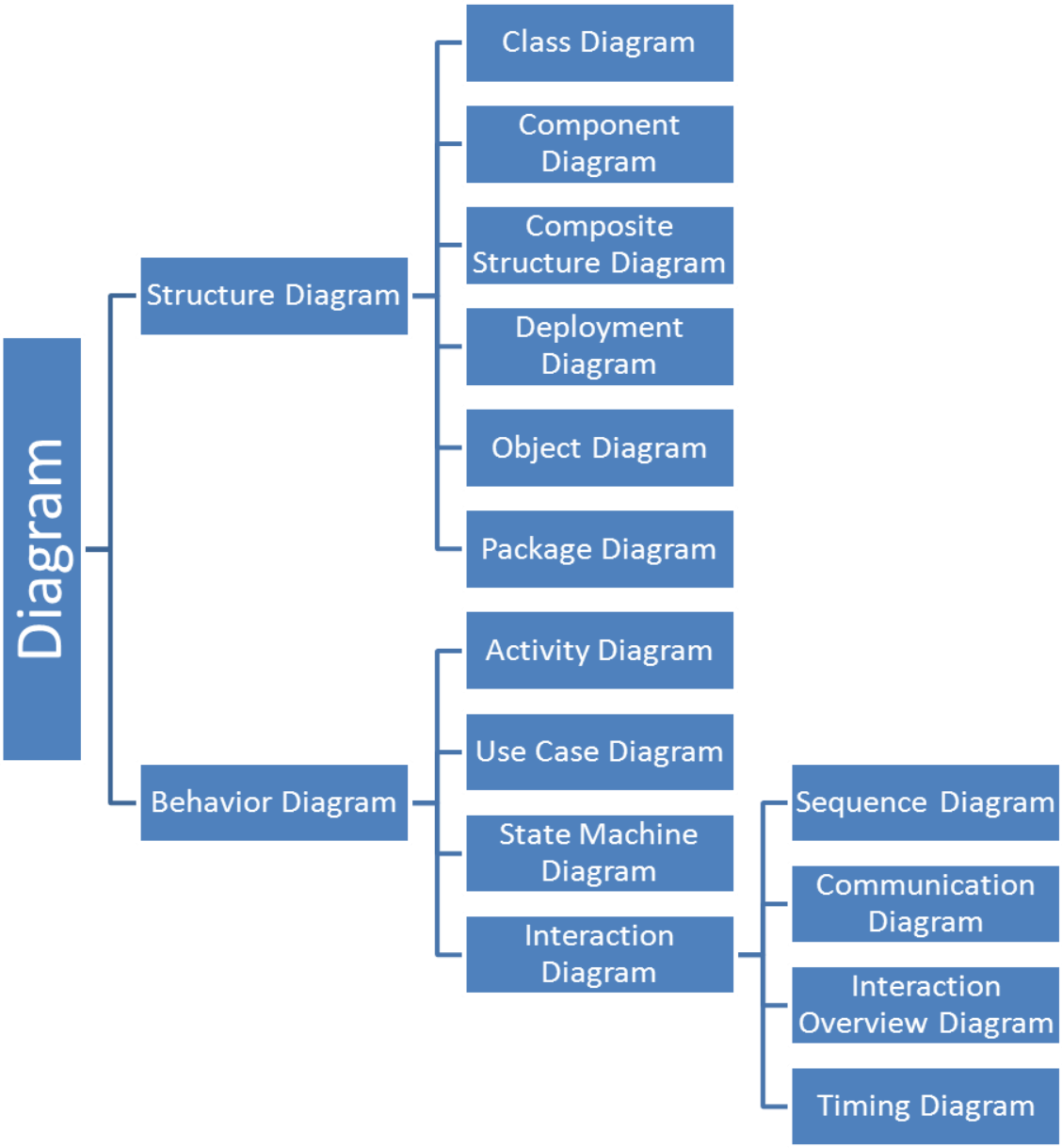
UML assists in detail design of complicated software systems through offering systematic and well understood babel with the developers, analysts, and the stakeholders. Thumbs up for it, as it accommodates the object-oriented ideas and is also good at practicing the number one rules of software engineering and is a good tool for planning, structuring, and organizing the software projects.

5.1 - UML DIAGRAMS:

UML is a modelling language that is platform agnostic and can be used with any programming language and with any sort of development methodology and because it is a standard most software developers will be familiar with it. That is to say, although many engineers may not prefer the use of diagrams, they are useful in an Agile development environment as they gradually add value while contributing positively to the flow and pace of work. It will be helpful if you treat UML diagrams not only as additional, and rather fancy, artifacts but as part of documentation. UML diagrams can assist engineering teams in several ways: UML diagrams can assist engineering teams in several ways:

- To welcome a new member on the team or an experienced developer, who joined a new team at the company.
- How to effectively behave for source code.
- This involves coming up with new features in a project before the programming phase of the project is started.
- Enabling easier understanding of the information where there may be an interaction with both the technical and non-technical observers.

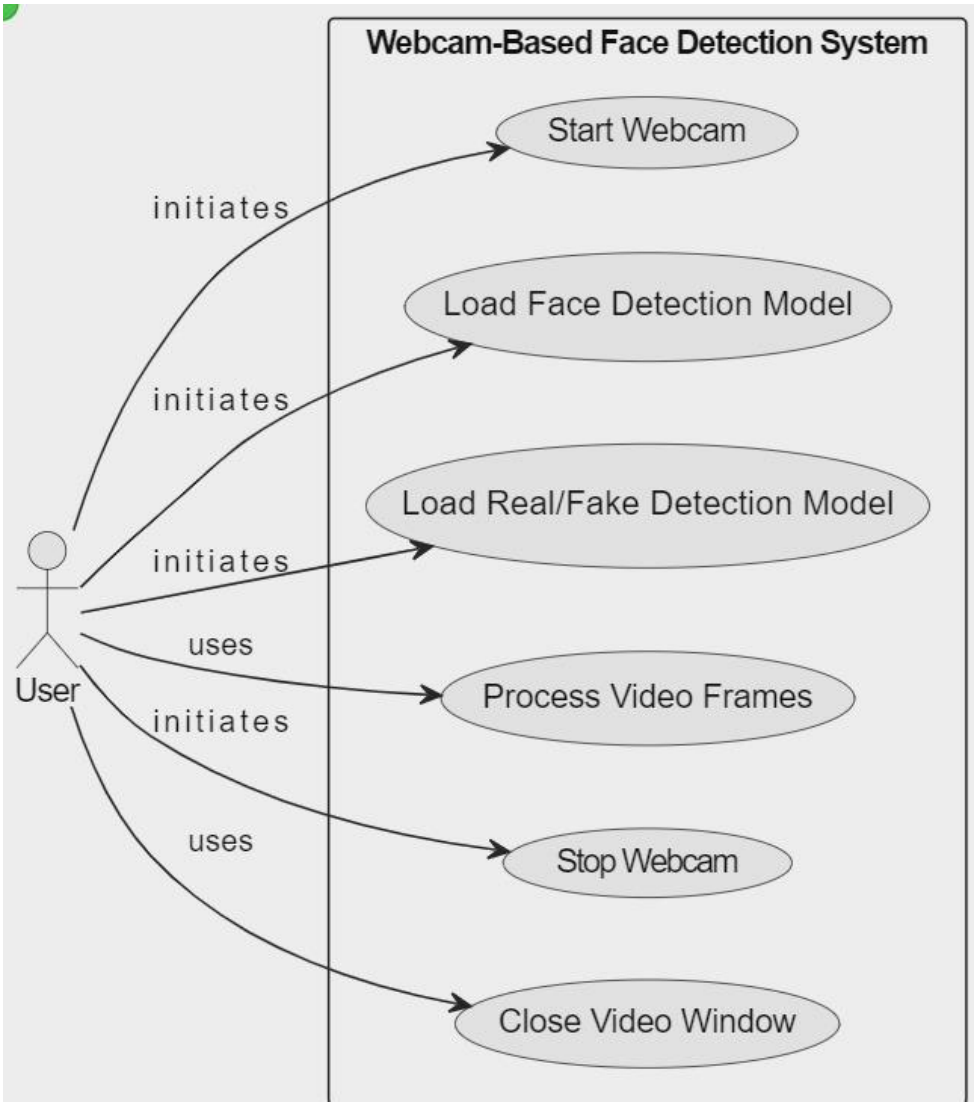
- Diagrams in UML can be broadly classified…



[Fig -7]: Concepts of UML

5.2 Use case Diagram:

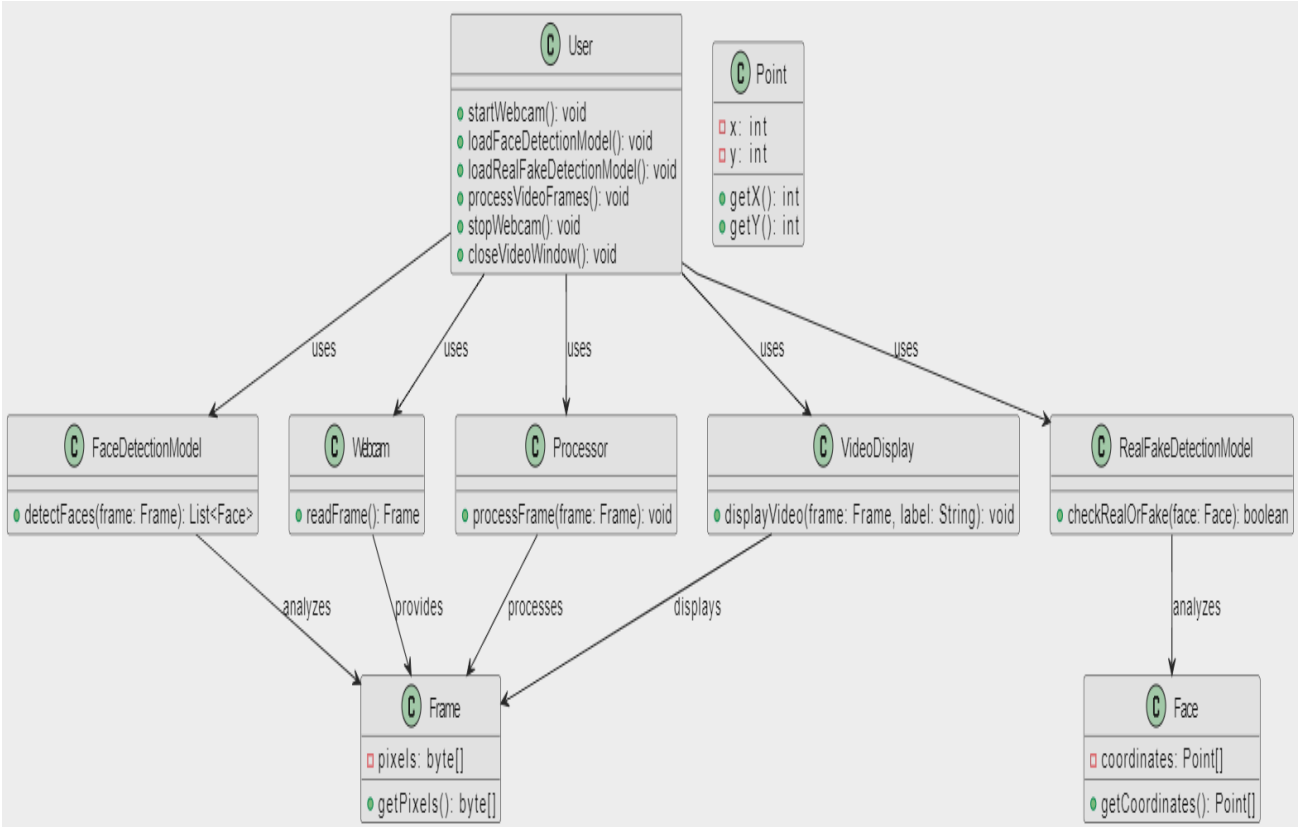
The use case diagrams create a picture view of the actors active in software program tool.
As structures that show device skills, they provide help builders contemplate how use instances relate to personas and define what a device is expected to do.



[Fig - 8] Use Case Diagram

5.3 Class Diagram:

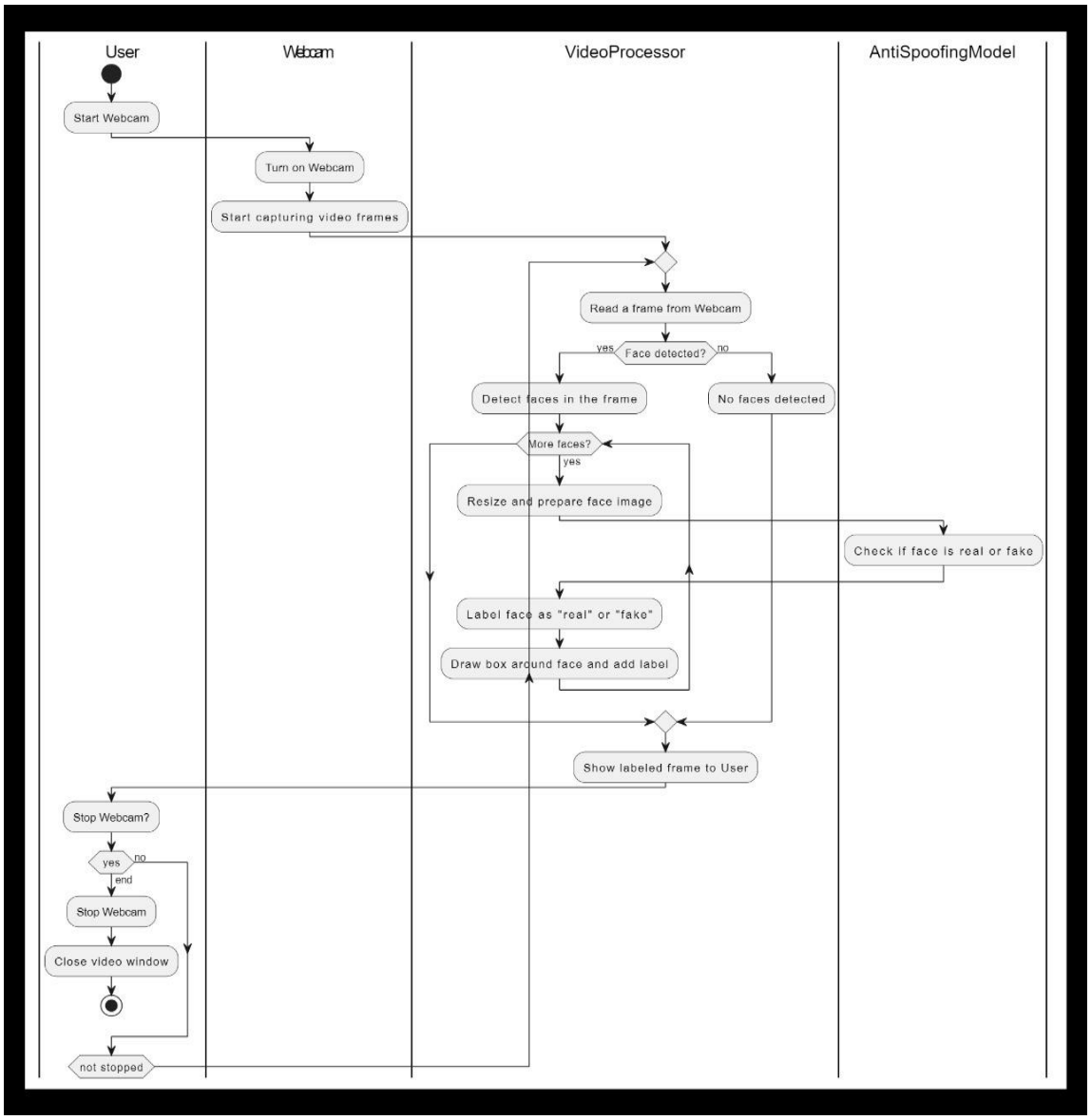
A UML elegance diagram is an element of a fundamental schema of any object-orientated option. This sketch represents a rigid thing-orientated machine, which defines initiatives supported by using schooling, properties and procedures. In other words, it denotes the teaching in a machine and the working of each one.



[Fig – 9] Class Diagram

5.4 Activity diagram:

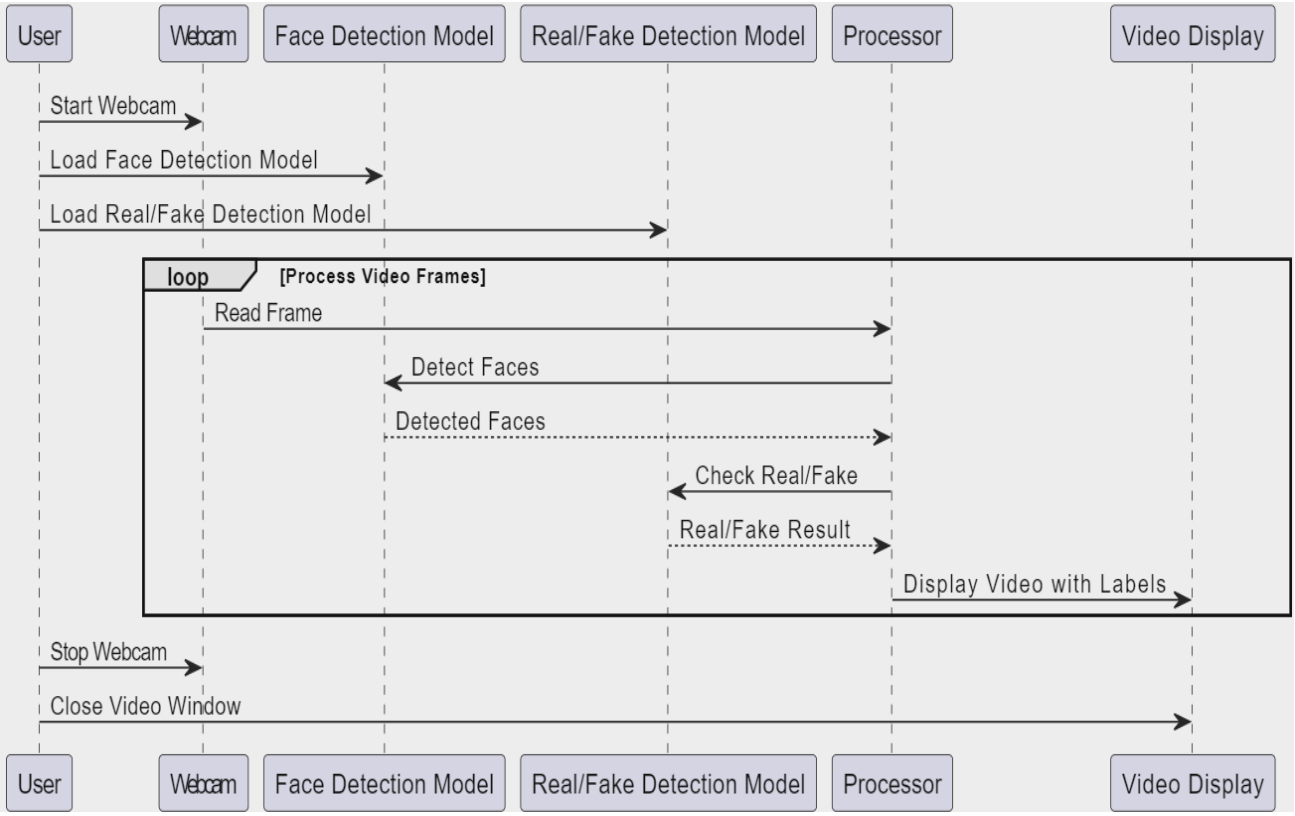
UML activity diagram is a flow chart that will display all the activity of a given system. It proposes the entire thing from start to end, outlining the demanding situations and the distinct selection routes and stages that appear to look like entire from one hobby to every other. The steps may be arranged in time, as a timeline, branches from a timeline, or parallel.



[Fig – 10] Activity Diagram

5.5 Sequence Diagram:

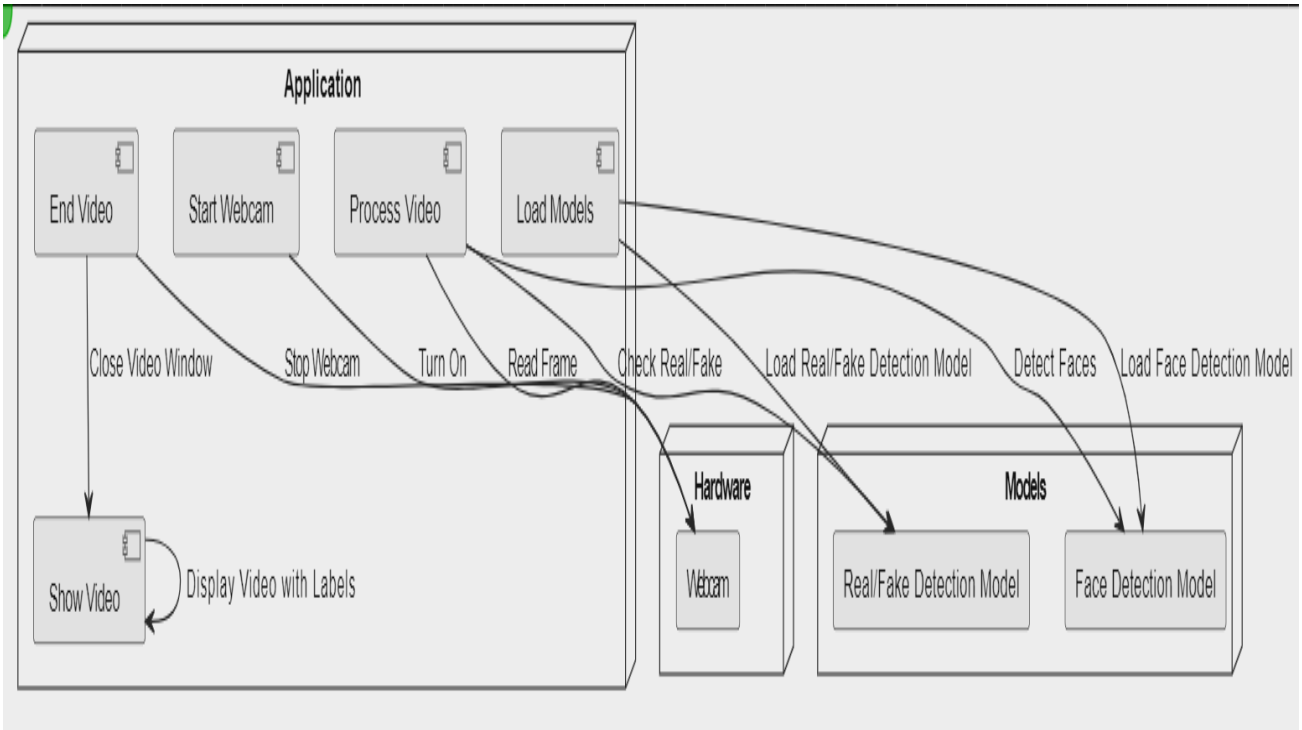
Another of the UML diagrams of collection is the collection sequence diagrams also frequently known as the occurrence diagrams that present the sequential flow of your gadgets. This includes the very threads of your character pieces and the schemes and dialogues of characters with your gizmos along with the conversations amongst these gizmos to fulfil a particular purpose.



[Fig – 11] Sequence Diagram

5.6 Deployment Diagram:

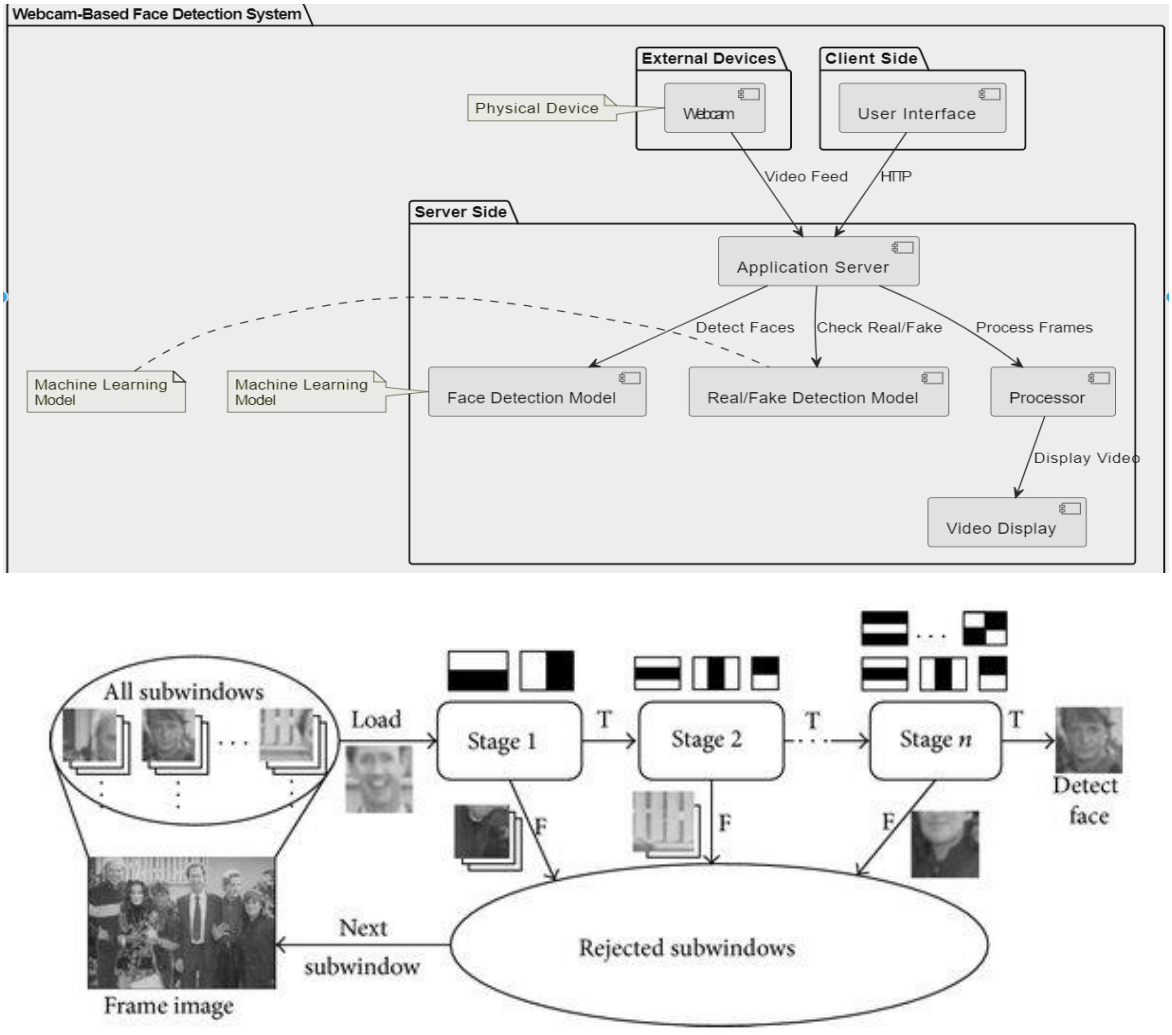
A deployment diagram show the envisioned distribution of nodes and all the additions that remain there on. In its simplest form, it portrays how the nodes in a distributed environment are joined physically. These diagrams are mainly helpful during the time of designing software that should effectively work in distinct physical platforms.



[Fig – 12] Deployment Diagram

5.7 Architecture

Convolutional Neural Networks (CNNs) are important and required for face anti-spoofing because they can effectively distinguish between spoofing and real faces (for example, pictures, films, and masks). This is through acquisition of hierarchical features from input images in order to determine complex patterns and textural formations associated with real faces and faces that have been synthesised. The fisher's linear discriminant classifier and Convolutional, Fully connected and pooling layers are crucial components. In general, performance is raised using advanced approaches such as transfer learning and deep CNNs. To elaborate, strong face anti-spoofing systems need to recognize features such as skin, tone, and reflection, and that CNNs are adept in analyzing due to their capability to analyze spatial pyramids.



[Fig – 13] System Architecture

6. IMPLEMENTATION

Sample Data for the Problem Statement:

A face anti-spoofing dataset contains a variety of facial representations, including printed pictures, digital displays, 3D masks, and actual human faces. They cover a range of lighting conditions, viewpoints, and facial expressions to effectively train and assess the system's ability to discern between real and fake faces.

[Fig – 14] Sample Data from the Dataset

Model Construction

Mount Google Drive and Unzip the Dataset

- Mounts Google Drive.
- Copies the dataset zip file from Google Drive to the local file system.
- Extracts the contents of the zip file.

```
[2] from google.colab import drive
    drive.mount('/content/gdrive')

    Mounted at /content/gdrive

[3] !cp -r "/content/gdrive/MyDrive/final_antispoofing.zip" "/content"

[4] import zipfile
    archive = zipfile.ZipFile('/content/final_antispoofing.zip')
    archive.extractall('/content')
```

[Fig – 15] Mounting google drive

Prepare Directory Structure for Processed Dataset

- Defines paths to the dataset and its train/test directories.
- Creates a new directory structure to organize the processed dataset.

```
[6] dataset_dir = '/content/final_antispoofing'
    train_dataset_dir = '/content/final_antispoofing/train'
    test_dataset_dir = '/content/final_antispoofing/test'

[7] import os
    os.mkdir('/content/antispoofing_dataset')
    os.mkdir('/content/antispoofing_dataset/train')
    os.mkdir('/content/antispoofing_dataset/test')
    os.mkdir('/content/antispoofing_dataset/train/real')
    os.mkdir('/content/antispoofing_dataset/train/spoof')
    os.mkdir('/content/antispoofing_dataset/test/real')
    os.mkdir('/content/antispoofing_dataset/test/spoof')

[8] train_dir='/content/antispoofing_dataset/train'
    test_dir = '/content/antispoofing_dataset/test'
```

[Fig -16] Organizing the dataset

Split Dataset into Train and Test Sets

Defines a function to split the dataset into train and test sets, copying images to the appropriate directories.

```
[9]  import shutil
     import numpy as np
     import matplotlib.pyplot as plt
     import cv2

[10] def train_test_splits(data_directory):
       for split_type in os.listdir(data_directory):
         path_to_split_type = os.path.join(data_directory,split_type)
         for category in os.listdir(path_to_split_type):
           path_to_category = os.path.join(path_to_split_type,category)
           for subject in os.listdir(path_to_category):
             path_to_subject = os.path.join(path_to_category,subject)
             for img in os.listdir(path_to_subject):
               if split_type == 'train':
                 shutil.copy(os.path.join(path_to_subject,img),os.path.join(train_dir,category,img))
               else:
                 shutil.copy(os.path.join(path_to_subject,img),os.path.join(test_dir,category,img))

[11] train_test_splits(data_directory=dataset_dir)
```

[Fig – 17] Splitting the dataset

Explore and Print Dataset Statistic

```
categories = ['real','spoof']

[13] print("---------------------Exploring Training Datasets--------------------")
     for category in categories:
       path = os.path.join(train_dir,category)
       if category == 'real':
         r1 = len(os.listdir(path))
       else:
         s1 = len(os.listdir(path))
       print("There are {} images in {} directory".format(len(os.listdir(path)),category))
     print("There are {} total images in training directory".format(r1+s1))

     print("----------------------Exploring Testing Datasets-------------------------")
     for category in categories:
       path = os.path.join(test_dir,category)
       if category == 'real':
         r2 = len(os.listdir(path))
       else:
         s2 = len(os.listdir(path))
       print("There are {} images in {} directory".format(len(os.listdir(path)),category))
     print("There are {} total images in testing directory".format(r2+s2))
```

Prints the number of images in each category (real and spoof) for both training and testing datasets.

```
---------------------Exploring Training Datasets--------------------
There are 2102 images in real directory
There are 2118 images in spoof directory
There are 4220 total images in training directory
----------------------Exploring Testing Datasets-------------------------
There are 477 images in real directory
There are 474 images in spoof directory
There are 951 total images in testing directory
```

[Fig – 18] Printing dataset statistics

Visualize Samples from the Dataset

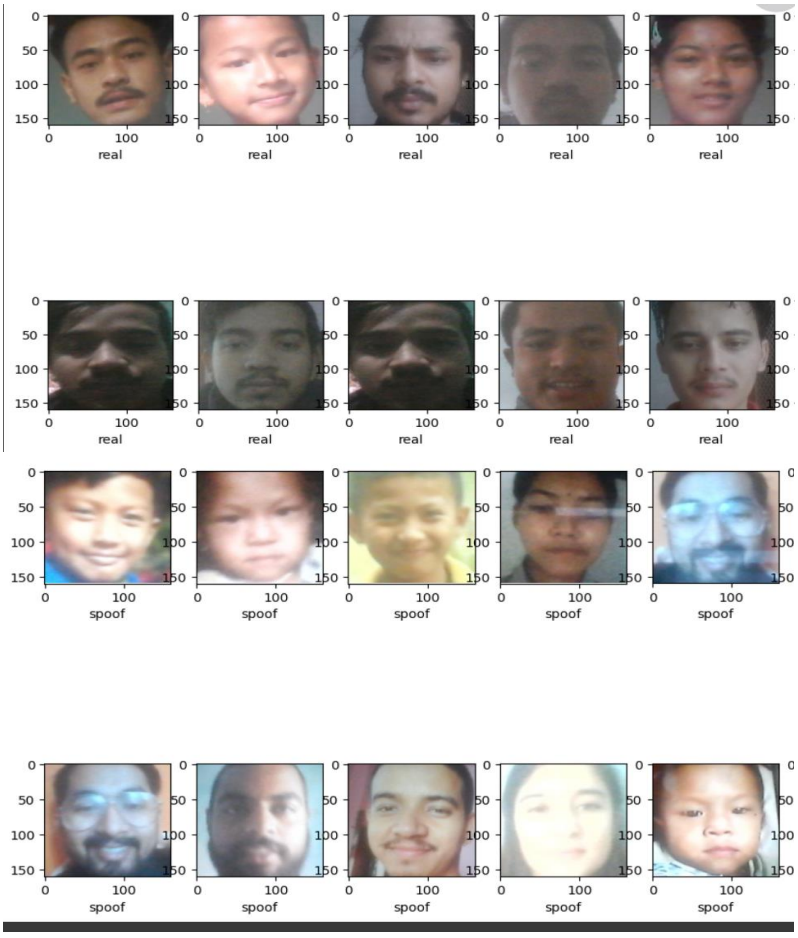
Defines functions to get image paths and visualize them.

```
def get_images(data_dir,number_of_samples):
  image_path = []
  for category in categories:
    path = os.path.join(data_dir,category)
    i = 1
    for img in os.listdir(path):
      if i > number_of_samples:
        break
      else:
        image_path.append(os.path.join(path,img))
        i += 1
  return image_path
```

```
def visualize_dataset(image_path,rows,cols):
  fig = plt.figure(figsize=(20,20))
  for i in range(1,rows * cols + 1):
    fig.add_subplot(rows,cols,i)
    img_array = cv2.imread(image_path[i-1])
    fig.subplots_adjust(hspace=1)
    plt.imshow(cv2.cvtColor(img_array,cv2.COLOR_BGR2RGB))
    plt.xlabel(image_path[i-1].split('/')[-2])
  plt.show()
```

```
training_image_path = get_images(data_dir= train_dir,number_of_samples=25)
print(training_image_path)
print(len(training_image_path))
```

[Fig – 19] Printing dataset statistics



[Fig – 20 & 21] Visualizing samples from both training and testing datasets

Create and Train the Model

- Creates data generators for training and validation datasets.
- Defines and compiles a MobileNetV2-based model.
- Trains the model, saving the best weights.

```
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint
from keras.applications.mobilenet_v2 import MobileNetV2

train_datagen = ImageDataGenerator(brightness_range=(0.8,1.2), rotation_range=30, width_shift_range=0.2, height_shift_range
valid_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir, target_size=(160,160), color_mode='rgb', class_mode='binary'
valid_generator = valid_datagen.flow_from_directory(test_dir, target_size=(160,160), color_mode='rgb', class_mode='binary',

mobilenet = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(160,160,3)))
mobilenet.trainable = False
output = Flatten()(mobilenet.output)
output = Dropout(0.3)(output)
output = Dense(units=8, activation='relu')(output)
prediction = Dense(1, activation='sigmoid')(output)

model = Model(inputs=mobilenet.input, outputs=prediction)
model.summary()

model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.000001, beta_1=0.9, beta_2=0.999, epsilon=1e-07),

os.mkdir('/content/model_weights/')
model_checkpoint = ModelCheckpoint('./model_weights/finalyearproject_antispoofing_model_{epoch:02d}-{val_accuracy:.6f}.h5',

history = model.fit_generator(train_generator, steps_per_epoch=train_generator.samples // 25, validation_data=valid_generat
```

[Fig – 22] Adding more epochs to the baseline model

Save the Model to JSON

```
[ ]  # serialize model to JSON
     model_json = model.to_json()
     with open("finalyearproject_antispoofing_model_mobilenet.json", "w") as json_file:
         json_file.write(model_json)
```

[Fig - 23]

Plot Training and Validation Accuracy and Loss

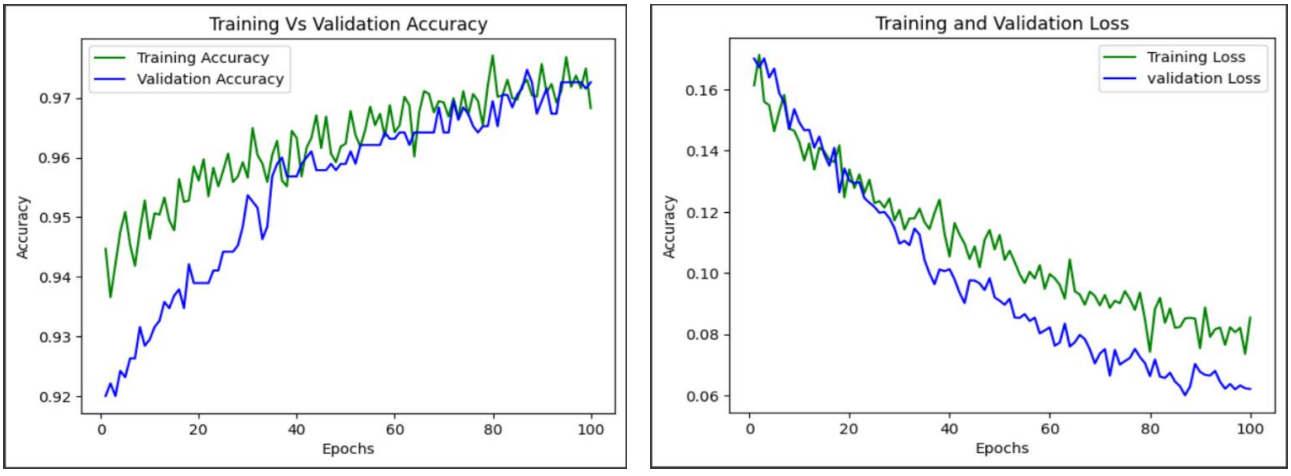
```
import matplotlib.pyplot as plt

train_accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']
epochs = range(1,101)
plt.plot(epochs, train_accuracy, 'g', label='Training Accuracy')
plt.plot(epochs, validation_accuracy, 'b', label='Validation Accuracy')
plt.title('Training Vs Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

train_loss = history.history['loss']
val_loss = history.history['val_loss']
plt.plot(epochs, train_loss, 'g', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

[Fig - 24]

Plots of the training and validation accuracy and loss over epochs.



[Fig – 25] Output of the above plotting

Evaluate Model on Test Set

```
from keras.preprocessing import image
import numpy as np
def check_fakes(path,category_type):
  predictor = {}
  path= os.path.join(path,category_type)
  for img in os.listdir(path):
    try:
      img = image.load_img(os.path.join(path,img),target_size=(160,160))
      img = image.img_to_array(img)
      img = np.expand_dims(img,axis=0)
      img = img / 255.0
      prediction = model.predict(img)
      if prediction > 0.5:
        prediction_class = 1
      else:
        prediction_class = 0
      result = categories[prediction_class]
      if result not in predictor:
        predictor[result] = 1
      else:
        predictor[result] += 1
    except Exception as e:
      pass
  return predictor

[ ] check_fakes(test_dir,categories[1])
```

[Fig – 26] Evaluating the model

Accuracy:

```
1/1 [==============================] - 0s 21ms/step
{'real': 475, 'spoof': 2}

[2]  accuracy = 100 - (19+6)/(19+455+471+6)*100

[3]  accuracy

97.37118822292324
```

[Fig - 27]

Anaconda Prompt:

```
Administrator: Anaconda Prompt

(base) C:\Windows\System32>cd C:\Users\SRIRAM\OneDrive\Desktop\Minor Project\Face_Antispoofing_System

(base) C:\Users\SRIRAM\OneDrive\Desktop\Minor Project\Face_Antispoofing_System>conda activate antispoofing

(antispoofing) C:\Users\SRIRAM\OneDrive\Desktop\Minor Project\Face_Antispoofing_System>python livelines_net.py
```

[Fig – 28]

7. SOFTWARE TESTING

Software testing, as defined, is a type of testing that is done before actual software is fully run. The primary purpose of testing the software is the outcome which is expected to meet requirements with limited or no errors or defects.

7.1 Unit Testing:

Haar Cascade Algorithm Components:

Objective: To experiment with Face detection deep under different dispensations.

Tests:

- Attempting to evaluate the algorithm's potential to draw a face from much light and little light environment (e. g. , bright, dim).
- Thankfully, its overall performance should be verified effectively with the different face positions and directions (frontal, profile, etc.).
- Correct identification is possible during wear in occlusions such as glasses or hat.

Convolutional Neural Network (CNN) Model: Convolutional Neural Network (CNN) Model:

Objective: Confirm that the version well categorizes in between actual and fake faces.

Tests:

- Check char layers and features in unit check section of the CNN version
- Ensure that mastering skills and results of the chosen version actually match the version,
- along with certain general performance-related parameters, such as precision and recall to name a few.
- Conclusively, attempt resiliency towards quite common adversarial attacks on face recognition techniques.

7.2 Integration Testing:

Integration of Haar Cascade and CNN:

Objective: Assure the flow and interaction between the face detection and popularity subscripts are smooth and integrated.

Tests:

- Truly address the identification of face areas using way of means of way Haar Cascade as enter toward the CNN model.
- Validate the general device accuracy and performance in detecting and verifying faces.

7.3 Functional Testing:

Functionality Across Environments:

Objective: Evaluate the machine overall performance with reference to its effectiveness beneath various environmental requirements.

Tests:

- Assess face detection and popularity underneath different lighting concerns (Natural light, Artificial light).
- Assess the performance of the machine in relation to differences in facial poses and orientations.
- Review the overall performance in different settings such as with glasses or with partial face occlusion.

❖ Performance Testing:

Objective: Measure machine reaction instances and aid utilization.

Tests:

- Benchmark of the processing speed of face detection and similarity match recognition tasks.
- When the actual operation is concerned, individuals need to know the amount of CPU and reminiscence utilization.
- Conduct pressure checks to assess machine overall performance beneath load.

7.4 Security Testing:

Spoofing Detection:

Objective: Assess the system`s resilience in opposition to spoofing attacks.

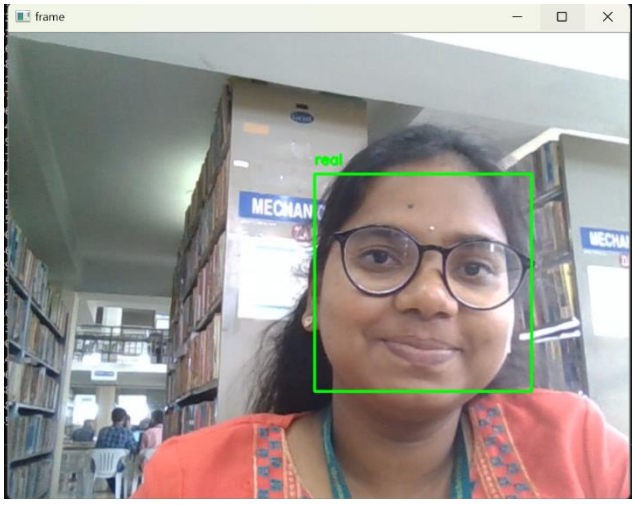
Tests:

- Test detection accuracy in opposition to spoofing tries the use of pics or videos.
- Evaluate the system's capacity to apprehend and reject faux faces (e.g., masks, revealed pics)

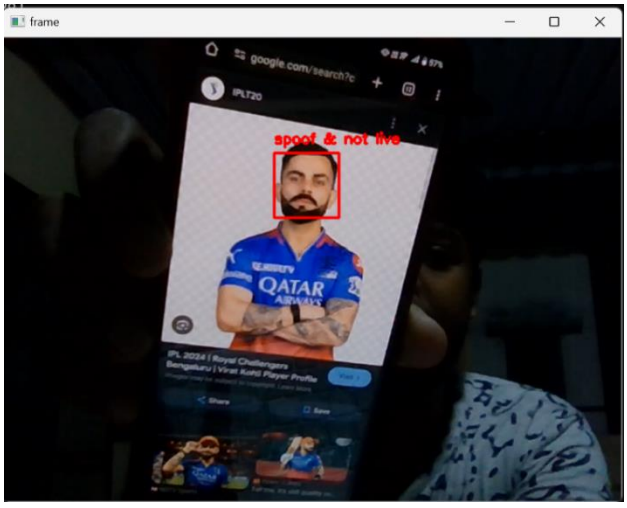
7.5 Testing on our System:

I tested face anti-spoofing with the use of the Haar Cascade algorithm, and the outcomes have proven that the system has high accuracy in determining genuine faces and successful thwarting attempts at spoofing the system. To my knowledge, the results corroborate the effectiveness of the system, testifying to its stability, efficiency, and applicability in real environments to improve security and authorization functions.

8. RESULTS



Real for a live person



Spoof for an image on phone

9. CONCLUSION AND FUTURE ENHANCEMENTS

The Haar Cascade algorithm of face anti-spoofing integrated with the CNN achieves promising performances while differentiating the face from its spoofs. Haar Cascade performs a fast method in the initial step of face detection and afterward, we utilize Convolutional Neural Network as a computer vision approach in real-time that uses deep learning algorithms. This dual-pronged approach enhances security system robustness against such attacks more effectively since other strategies employed were constantly thwarted by spoofing attempts.

Future events may focus on different aspect which could involve a variety of options.

- First, improving the features of a CNN structure to handle complex spoofing techniques.
- Second, the use of multiple biometric authentication modes (e. g. iris or face recognition in addition to each other) for making systems safer.
- Third, optimizing the speed of real-time performance based on enhancing the framerates for the hardware and effectiveness for the algorithms used.

This is achievable by altering the type of spoofing material that is used, different lighting conditions, as well as different ethnic backgrounds of the subjects. And last but not the least, last action studying convenient learning methods to track spoofing strategies and update the system regularly. With these improvements, the system will be more dependable and acceptable in a number of real world conditions and it will provide an overall protection against face spoofing attacks.

10. BIBLIOGRAPHY

[1] Face Recognition With Liveness Detection Using Eye And Mouth Movement ;
A. Singh, P. Joshi, G.C. Nandi

[2] Anti-Spoofing In Face Recognition With Liveness Detection Using Pupil Tracking ;
M. Killioğlu; M. Taşkiran; N. Kahraman

[3] A Review Spoof Face Recognition Using Lbp Descriptor ;
Tanvi Dhanwanpatil & Bela Joglekar

[4] Face Spoof Detection System Based On Genetic Algorithm And Artificial Intelligence Technique ; D. ANAND & K. GUPTA

[5] The Effectiveness Of Depth Data In Liveness Face Authentication Using 3d Sensor Cameras ; Ghazel Albakri & Sharifa Alghowinem

[6] A New Ensemble Of Texture Descriptors Based On Local Appearance-Based Methods For Face Anti-Spoofing System ; G.Thippeswamy, H. Vinutha, and R. Dhanapal

[7] Convolutional Neural Networks For Face Anti-Spoofing And Liveness Detection ;
Hsueh-Yi Sean Lin and Yu-Wei Su

[8] Face Spoofing Detection Using Enhanced Local Binary Pattern;
Karuna Grover, Rajesh Mehra

iThenticate

Similarity Report ID: oid:3618:62998573

18% Overall Similarity

Top sources found in the following databases:

- 12% Internet database
- 1% Publications database
- Crossref Posted Content database
- 13% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	**Educational Service District 105 on 2022-04-16** Submitted works	5%
2	**Fırat Üniversitesi on 2021-08-02** Submitted works	4%
3	**ijraset.com** Internet	2%
4	**coursehero.com** Internet	1%
5	**grietinfo.in** Internet	<1%
6	**gecgudlavalleru.ac.in** Internet	<1%
7	**res.ijsrst.com** Internet	<1%
8	**9pdf.net** Internet	<1%
9	**Symbiosis International University on 2021-09-22** Submitted works	<1%

Sources overview