DAYANANDA SAGAR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY
KUDLU GATE
BANGALORE - 560068

MAJOR PROJECT REPORT

ON

"Spoofed Face Detection"

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by

Y. Anoop Reddy(ENG16CS0198)
Shivam Rungta (ENG16CS0153)
Srikanth Shastry (ENG16CS0162)

Under the supervision of
Professor Karishma Chavhan

DAYANANDA SAGAR UNIVERSITY

School of Engineering, Kudlu Gate, Bangalore-560068

CERTIFICATE

This is to certify that Shivam Rungta (ENG16CS0153), Srikanth Shastry (ENG16CS0162), Y. Anoop Reddy (ENG16CS0198) have satisfactorily completed their Major Project Phase III as prescribed by the University for the 8th semester B.Tech. programme in Computer Science & Engineering during the year 2019-2020 at the School of Engineering, Dayananda Sagar University, Bangalore.

Date: _____

Prof Karishma Chavhan	Dr. M.K Banga
Professor	Chairman
Department of Computer Science and Engineering	Department of Computer Science & Engineering

DECLARATION

We hereby declare that the work presented in this major project entitled "**SPOOFED FACE DETECTION**" submitted to **DAYANANDA SAGAR UNIVERSITY, Bengaluru** is a record of group work implemented under the guidance of **Mrs. Kariahma Chavhan,** Professor, Department of CSE, Dayananda Sagar University.

This project is submitted in the partial fulfilment of the requirements and conductive criticism for the award of VIII semester major-project**.** The results embodied in this project have not been submitted elsewhere for the same.

TABLE OF CONTENTS

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We are especially thankful to our **CHAIRMAN Dr. M K Banga,** for providing necessary departmental facilities, moral support and encouragement.

We are very much thankful to **Prof Karishma Chavhan,** Professor, Department of CSE, Dayananda Sagar University, for her helpful tips and suggestions in carrying out with major project successfully.

We have received a great deal of guidance and co-operation from our friends and we wish to thank one and all who have directly or indirectly helped us in the successful completion of this project.

TABLE OF CONTENTS

SPOOFED FACE DETECTION

INTRODUCTION

Consider what would happen if a nefarious user tried to purposelycircumvent your face recognition system.Such a user could try to hold up a photo of anotherperson. Maybe they even have a photo or video on theirsmartphone that they could hold up to the camera responsiblefor performing face recognition.In those situations it's entirely possible for the face held up to thecamera to be correctly recognized…but ultimately leading to anunauthorized user bypassing your face recognition system!How would one go about spotting these "fake" versus"real/legitimate" faces? How could you apply anti-face spoofingalgorithms into your facial recognition applications?

The answer is to apply spoofed face detection.

Some adavntages of using spoofed face detection is improved security, in specific by avoiding a nefarious user from accessing sensitive information, we can avoid possible data theft.

Areas of application would be during online exams to make the person taking the exam is a real person and not an image, use in projects making use of sensitive data or data that is not to be publicly disclosed.

PROBLEM DEFINITION

The problem here is to identify weather the person in frount of the screen is a real person or a spoofed replica of a person, this can be achived by using a spoofed face detecton algorithm. This done to restrict unwanted users from accessing content that uses facedetection as a access feature. Considering the case of a person trying to access/hack ones phone without his/her permisson or even trying to cheat an online face id verification he can do so by spoofing the face recognition biometrics by using a video or an image the owner of the phone and unlock it,the hacker will have unrestricted access to the owner's phone and in the other scenario is able to make someone else give the verification by cheating the system. This is a problem that may occur in anyscenario using face biometrics as a security feature.

LITERATURE SURVEY

1.Because of its natural and non-intrusive interaction, identity verification and recognition using facial information are among the most active and challenging areas in computer vision

research. Despite the significant progress of face recognition technology in the recent decades, wide range of viewpoints, ageing of subjects and complex outdoor lighting are still research challenges. Advances in the area were extensively reported recently.However, checking if the face presented to a camera is indeed a face from a real person and not an attempt to deceive (spoof) the system has mostly been overlooked. A spoofing attack consists of the use of forged biometric traits to gain illegitimate access to secured resources protected by a biometric authentication system. Recently, the media has documented some situations of attacks in deployed face recognition systems. Using simple photographs, a research group from University of Hanoi showed how easy is to spoof the face authentication systems deployed in Lenovo, Asus and Toshiba Laptops. Since the release Ice Cream Sandwich the Android OS comewitha built-in face authentication system to unlock the mobile phone. Since then, it has been extensively demonstrated around the web how easy it is to spoof this face recognition system. As a consequence, an eye blinking detection has been introduced in the most recent version of the Android OS. The lack of protection against biometric spoofing attacks is not exclusive to face biometrics. The finding indicate that fingerprint authentication systems suffer from a similar weakness. The same shortcomings on iris recognition systems have been diagnosed. While it is possible to spoof a face authentication system using make-up, plastic surgery or forged masks; photographs and videos are probably the most common threats. Moreover, due to the increasing popularity of social network websites (facebook, youtube, flickr, instagram and others), a great deal of multimedia content, specially videos and photographs, is available on the web that can be used to spoof a face authentication system. To mitigate vulnerabilities, effective countermeasures against face spoofing must to be deployed. It was not until very recently that the problem of spoofing attacks against face biometric systems gained attention of the research community. This can be attested by the gradually increasing number of publicly available databases, The JCB 2017 competition on to see, study and find countermeasures to 2D facial spoofing attacks.

2.Forgery of face images in selfies are increasing now days. The lack of efficient algorithm, authorities could not find the fake faces from these forged photographs. Here a novel algorithm is designed and implemented for online social media. This system consists of face detection for detecting human face present in the image, and then it is transformed into different color space model. Illuminant map is obtained for each detected face. Also image quality measure for detected face is estimated. Using these features it indentifies the input image as spliced or not. This algorithm tested on datasets DSO-1 and DSI-1, and gets an accuracy of 96.5%, sensitivity of 94.4% and specificity of 94.7%. This algorithm can be easily implemented in hardware based systems thus it makes biometric and forensic systems are more robust. Apart from the face images, this algorithm can be used to detect spliced objects from the scenes. The accuracy can be improved by adding some more image quality measures. The speed of the system can be improved by excluding the non-reference image measures.

OBJECTIVES

The objective of this project is to build a facial recognition system that is able to tell the difference between a regular normal face of a legid user from a fake spoofed one that is a video

or an image of the legid person and accordingly give the possiblity of it being a fake or a real live person thus avoiding access to undesired elements trying to get in the sytem by spoofing it with a fake video or image. This system will alert that the face biometrics has spotted a face and subsequently provied a number(0-0.9 fake/real) to estimate the degree oflegitimacy of the image or video and intern the user itself.

REQUIREMENT ANALYSIS

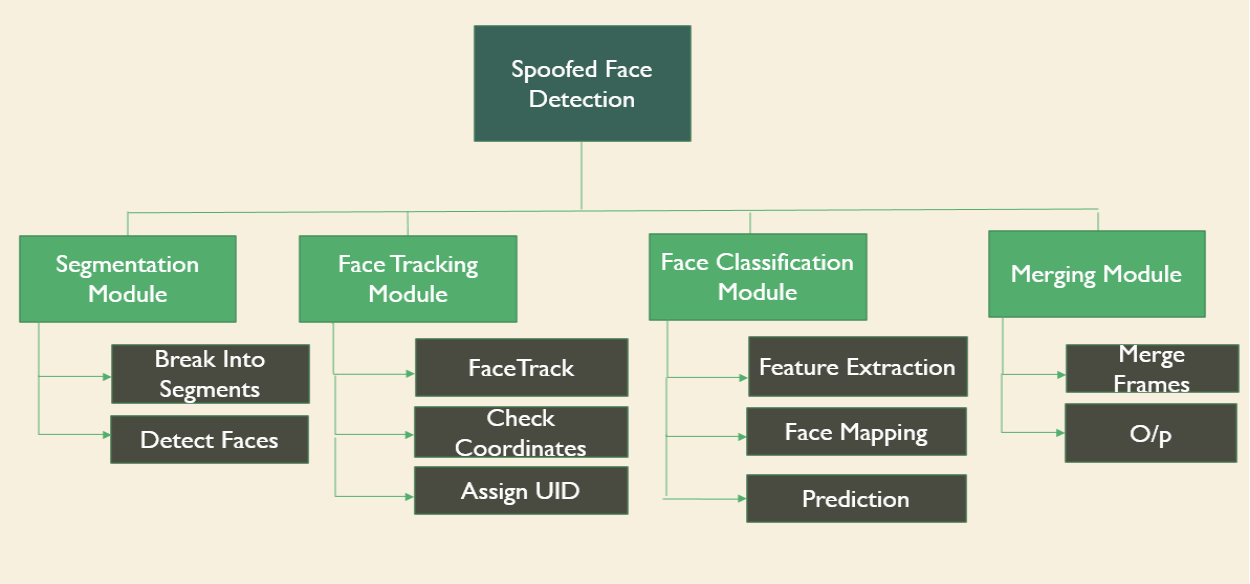
Software Requirements:

- OpenCV
- Keras
- Python3

Hardware requirements:

- **For Development**
  - Laptop / PC with minimum 2GB RAM
  - Windows / Linux (any Distro) OS
- **For Deployment**
  - Camera Module or USB Webcam

MODULE WISE BREAKDOWN



- **Segmentation Module**

-**Break into Segments**- Input received from the web camera are broken down into frames or segments

-**Detect Face-** Face detection is applied the Regions of Interest(ROIs) corresponding to each frame.
-Using openCV deeplearning face detector, CAFFE(Convolutional Architecture for Fast Feature Embedding).

- **Face tracking**

-**Face Track**- The face tracking module associates faces captured in successive frames to a same person and can also confirm ROIs found through segmentation.

-**Check coordinates** -The coordinates of one frame can be used to predict the possible coordinates of the face in the next frame.

-**Assign UID**- The tracker assigns a unique identifier for each face track so that the tracking module can follow the movement of a face in a sequence

-Heuristic-based algorithms, including eye movement, lip movement, and blink detection. These set of algorithms attempt to track eye movement and blinks, Optical Flow algorithms**,** namely examining the differences and properties of optical flow generated from 3D objects and 2D planes

- **Face classification**

-**Feature Extraction**- During operations, ROIs are detected in successive frames of a video, and then the same features are extracted into an input ROI pattern.

-**Face Mapping**- Mapping is done to each frame and its coordinates are compared the values of the next frame, computes similarity of input ROI patterns against standards enrolled to the system. Mapping is typically performed using a pattern classifier.

-**Prediction**- The results coordinates and the comparison of standards the likelihood that an input ROI captured being fake or real is predicted.

-Also uses caffe model of openCV, uses GoodLeNet training network**,** Texture analysis**,** including computing Local Binary Patterns (LBPs) over face regions and using an SVM to classify the faces as real or spoofed

- **Merging**

-**Merge Frames** – Individual frames are merged after the classification, mapping, prediction and output is displayed with probability of the face being fake or Real.

ARCHITECTURE DIAGRAM

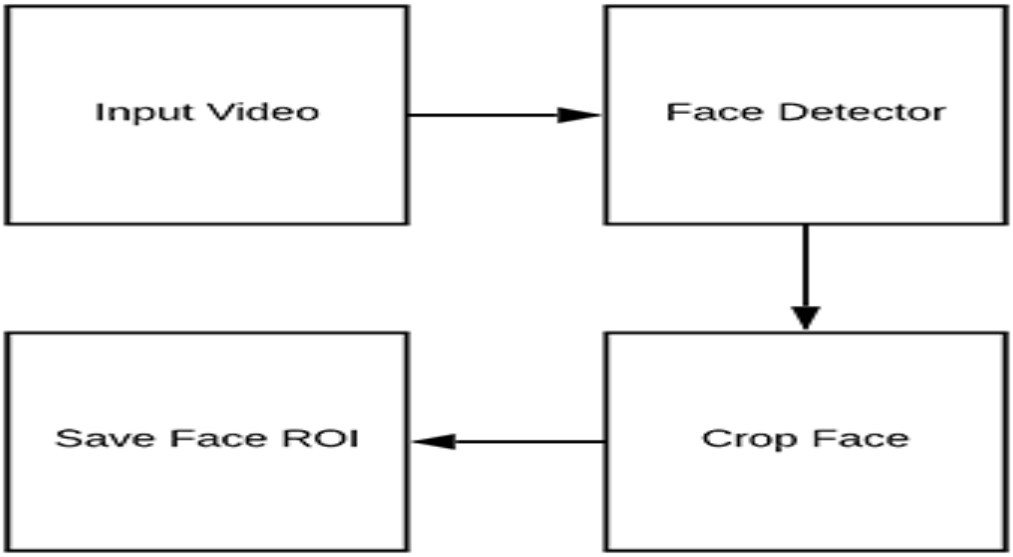


Fig.1 basic flow diagram

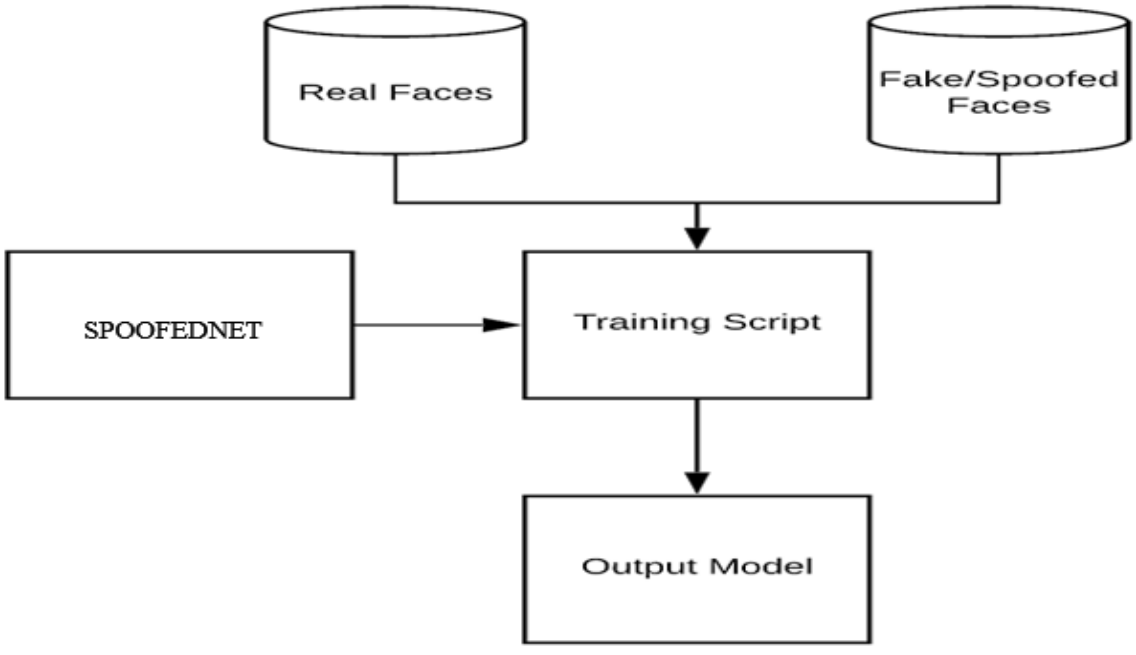


Fig.2 Basic training diagram

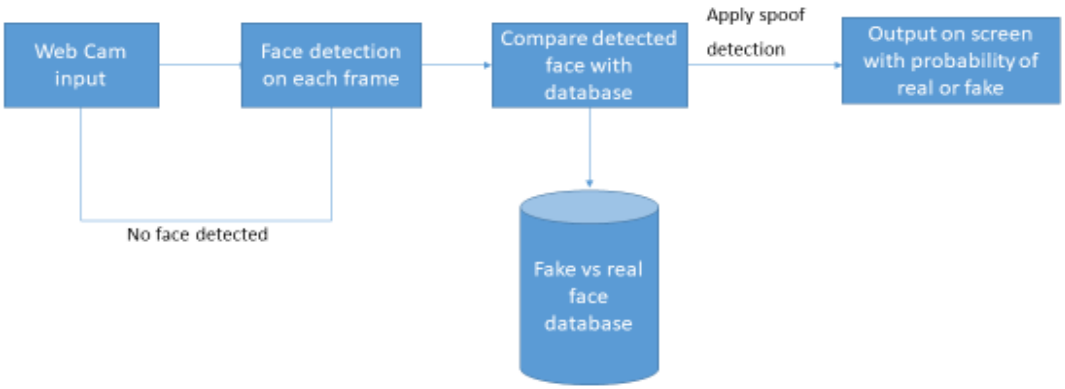


Fig.3 Basic architecture diagram.

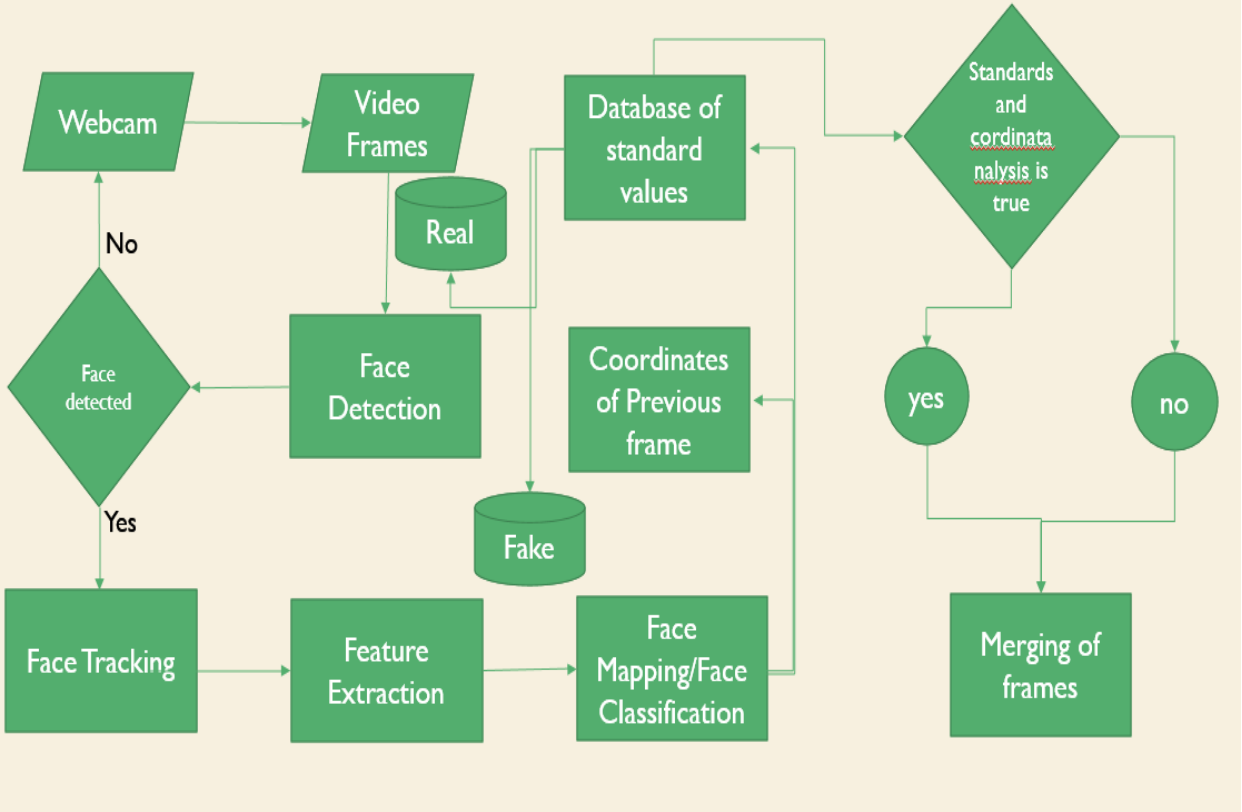


Fig.3 Detailed architecture diagram.

ALGORITHM

Input: Input frames of recorded video or realtime webcam output

Output: Display on screen with a box on the ROI with specification about the probability of the image/images being fake or real.

for each sample with frames= say k from 1,..., ∞ frames do

Apply segmentation on input video sample to detect ROIs corresponding to faces in a frame

//Detect Face

for each ROI from r =0,…,R do

if the ROI is located in a different place than the existing tracks then

Increment the number of tracks, K K+1

Use Heuristic-based algorithms for eye movement, lip movement, Compute a new Detect-face-model with the ROI for the initiated face track K

//Coordinate Analysis

for each Detect-face-model with k = 1,…,K do

Compute the coordinates of the face in frame using mapping

Update the Detect-face-model using the new state information.

Also use Texture analysis with local binary patters(LBPs).

//Classification

for each input frame associated with k=1,…,K do

for each image with standards with l=1,…,L do

Compute similarity between input frame and standard image

Compute Frequency analysis, Heuristic-based algorithms

and 3D face shape algorithm.

CODE IMPLEMENTATION

1. gather_examples.py

```
# USAGE
# python gather_examples.py --input videos/real.mov --output dataset/real --detector face_detector --skip 1
# python gather_examples.py --input videos/fake.mp4 --output dataset/fake --detector face_detector --skip 4

# import the necessary packages
import numpy as np
import argparse
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, required=True,
	help="path to input video")
ap.add_argument("-o", "--output", type=str, required=True,
	help="path to output directory of cropped faces")
ap.add_argument("-d", "--detector", type=str, required=True,
	help="path to OpenCV's deep learning face detector")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
	help="minimum probability to filter weak detections")
ap.add_argument("-s", "--skip", type=int, default=16,
	help="# of frames to skip before applying face detection")
args = vars(ap.parse_args())

# load our serialized face detector from disk
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
	"res10_300x300_ssd_iter_140000.caffemodel"])
```

```
net = cv2.dnn.readNetFromCaffe(protoPath, modelPath)

# open a pointer to the video file stream and initialize the total
# number of frames read and saved thus far
vs = cv2.VideoCapture(args["input"])
read = 0
saved = 0

# loop over frames from the video file stream
while True:
	# grab the frame from the file
	(grabbed, frame) = vs.read()

	# if the frame was not grabbed, then we have reached the end
	# of the stream
	if not grabbed:
		break

	# increment the total number of frames read thus far
	read += 1

	# check to see if we should process this frame
	if read % args["skip"] != 0:
		continue

	# grab the frame dimensions and construct a blob from the frame
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0,
		(300, 300), (104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the detections and
	# predictions
	net.setInput(blob)
	detections = net.forward()
```

```
# ensure at least one face was found
if len(detections) > 0:
    # we're making the assumption that each image has only ONE
    # face, so find the bounding box with the largest probability
    i = np.argmax(detections[0, 0, :, 2])
    confidence = detections[0, 0, i, 2]

    # ensure that the detection with the largest probability also
    # means our minimum probability test (thus helping filter out
    # weak detections)
    if confidence > args["confidence"]:
        # compute the (x, y)-coordinates of the bounding box for
        # the face and extract the face ROI
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")
        face = frame[startY:endY, startX:endX]
```

2. spoofedfacedetection.model.py

```
# USAGE
# python spoofedfacedetection_demo.py --model spoofedface.model --le le.pickle --detector face_detector

# import the necessary packages
from imutils.video import VideoStream
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import numpy as np
import argparse
import imutils
import pickle
import time
```

```
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--model", type=str, required=True,
	help="path to trained model")
ap.add_argument("-l", "--le", type=str, required=True,
	help="path to label encoder")
ap.add_argument("-d", "--detector", type=str, required=True,
	help="path to OpenCV's deep learning face detector")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
	help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector from disk
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
	"res10_300x300_ssd_iter_140000.caffemodel"])
net = cv2.dnn.readNetFromCaffe(protoPath, modelPath)

# load the spoofed face detector model and label encoder from disk
print("[INFO] loading spoofed face detector...")
model = load_model(args["model"])
le = pickle.loads(open(args["le"], "rb").read())

# initialize the video stream and allow the camera sensor to warmup
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# loop over the frames from the video stream
while True:
```

```
# grab the frame from the threaded video stream and resize it
# to have a maximum width of 600 pixels
frame = vs.read()
frame = imutils.resize(frame, width=600)

# grab the frame dimensions and convert it to a blob
(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0,
	(300, 300), (104.0, 177.0, 123.0))

# pass the blob through the network and obtain the detections and
# predictions
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in range(0, detections.shape[2]):
	# extract the confidence (i.e., probability) associated with the
	# prediction
	confidence = detections[0, 0, i, 2]
		endY = min(h, endY)

		# extract the face ROI and then preproces it in the exact
		# same manner as our training data
		face = frame[startY:endY, startX:endX]
		face = cv2.resize(face, (32, 32))
		face = face.astype("float") / 255.0
		face = img_to_array(face)
		face = np.expand_dims(face, axis=0)

		# pass the face ROI through the trained spoofed face detector
		# model to determine if the face is "real" or "fake"
		preds = model.predict(face)[0]
		j = np.argmax(preds)
```

```
                label = le.classes_[j]

                # draw the label and bounding box on the frame
                label = "{}: {:.4f}".format(label, preds[j])
                cv2.putText(frame, label, (startX, startY - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
                cv2.rectangle(frame, (startX, startY), (endX, endY),
                        (0, 0, 255), 2)

        # show the output frame and wait for a key press
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
```

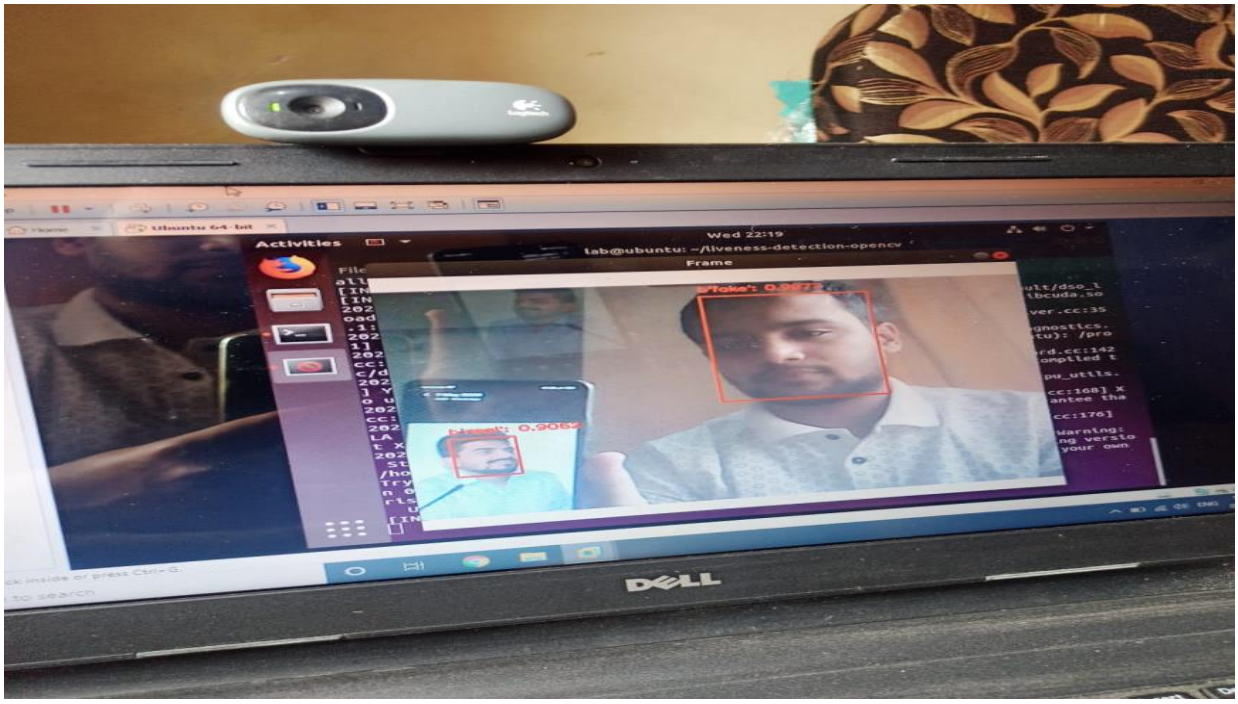
3. train_spoofedfacedetection.model.py

```
# USAGE
# python train_spoofedfacedetection.py --dataset dataset --model spoofedfacedetection.model --le
le.pickle


# set the matplotlib backend so figures can be saved in the background
import matplotlib
matplotlib.use("Agg")


# import the necessary packages
from pyimagesearch.livenessnet import LivenessNet
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.utils import np_utils
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
```

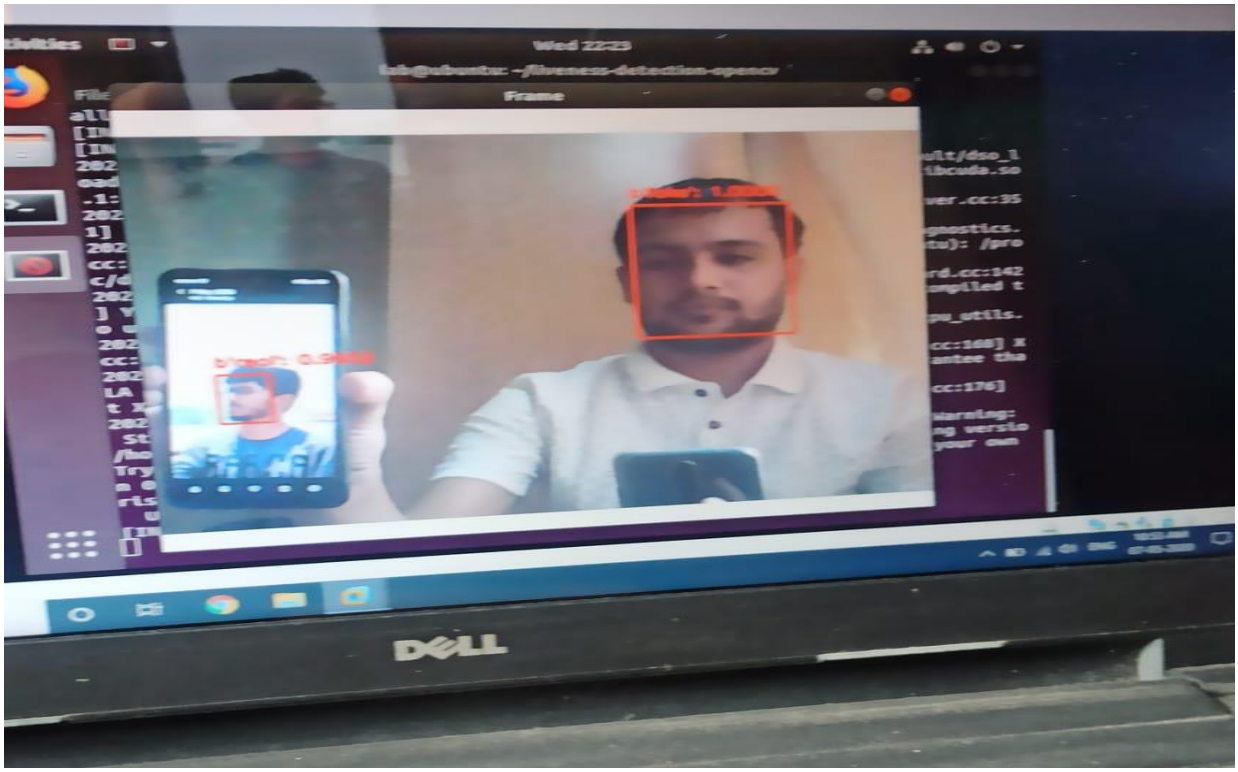
```
import argparse
import pickle
import cv2
import os

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []

for imagePath in imagePaths:
	# extract the class label from the filename, load the image and
	# resize it to be a fixed 32x32 pixels, ignoring aspect ratio
	label = imagePath.split(os.path.sep)[-2]
	image = cv2.imread(imagePath)
	image = cv2.resize(image, (32, 32))

	# update the data and labels lists, respectively
	data.append(image)
	labels.append(label)

# convert the data into a NumPy array, then preprocess it by scaling
# all pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") / 255.0

# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels,
	test_size=0.25, random_state=42)
```

```
# construct the training image generator for data augmentation
aug = ImageDataGenerator(rotation_range=20, zoom_range=0.15,
	width_shift_range=0.2, height_shift_range=0.2, shear_range=0.15,
	horizontal_flip=True, fill_mode="nearest")

# initialize the optimizer and model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model = LivenessNet.build(width=32, height=32, depth=3,
	classes=len(le.classes_))
model.compile(loss="binary_crossentropy", optimizer=opt,
	metrics=["accuracy"])

# train the network
print("[INFO] training network for {} epochs...".format(EPOCHS))
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=BS),
	validation_data=(testX, testY), steps_per_epoch=len(trainX) // BS,
	epochs=EPOCHS)
# evaluate the network
print("[INFO] evaluating network...")
predictions = model.predict(testX, batch_size=BS)
print(classification_report(testY.argmax(axis=1),
	predictions.argmax(axis=1), target_names=le.classes_))
# plot the training loss and accuracy
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, EPOCHS), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, EPOCHS), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, EPOCHS), H.history["acc"], label="train_acc")
plt.plot(np.arange(0, EPOCHS), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on Dataset")
```

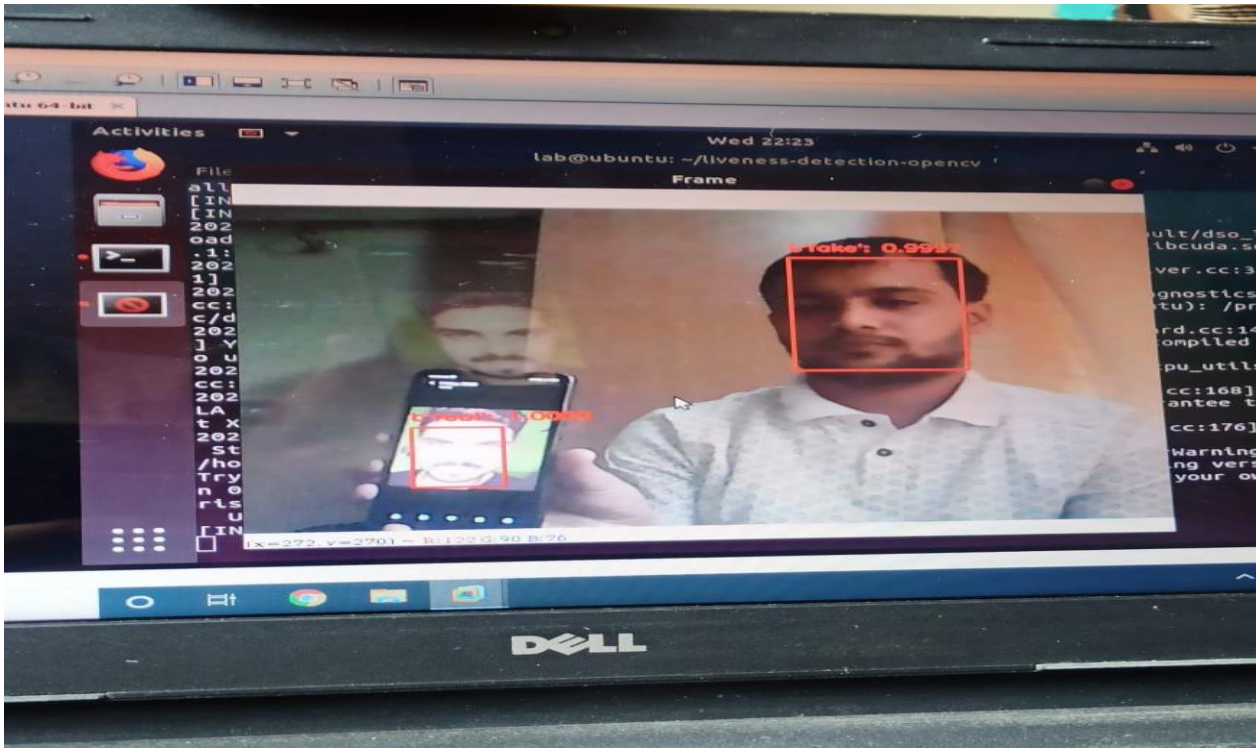
OUTPUT SCREENSHOTS



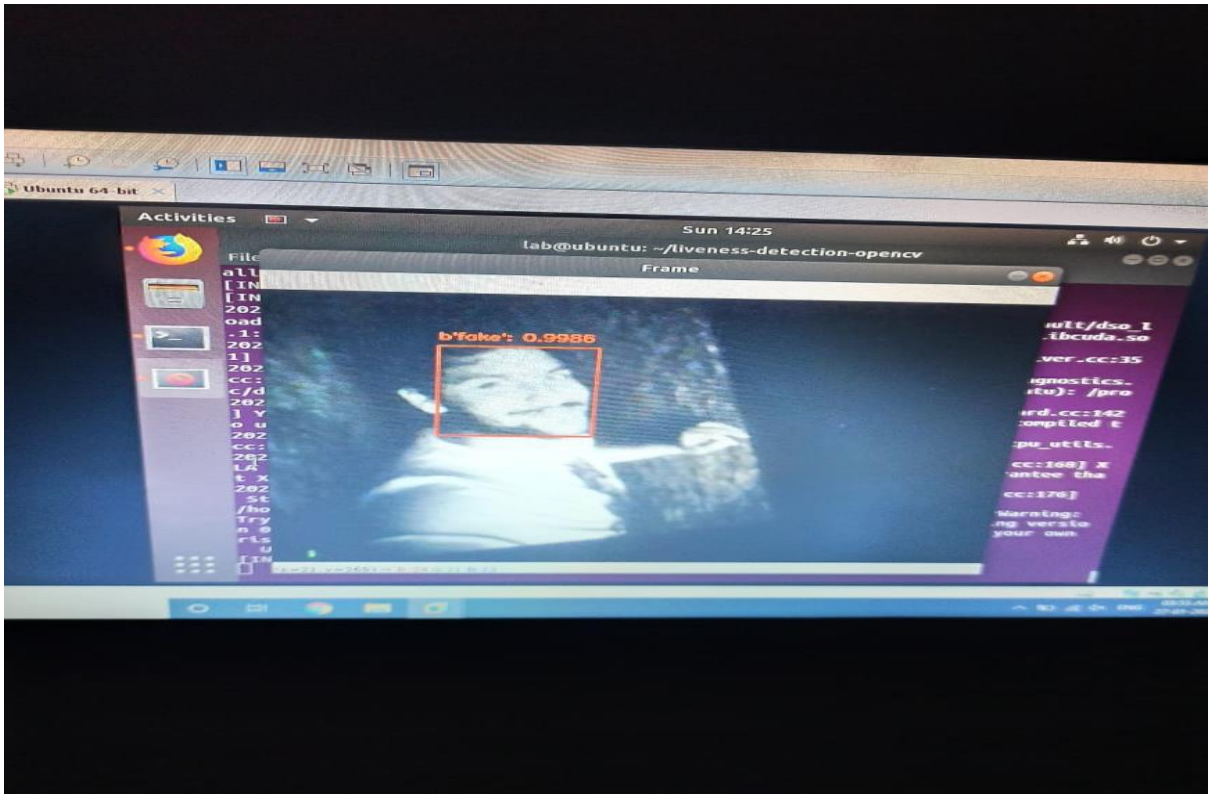
Teset case/Screen shot 1



Screen shot 2



Screen shot 3



Screen shot 4

FUTURE WORKS

Various anti-spoofing techniques are being developed and implemented that are helping in significantly raising the level of difficulty of spoofing attacks. Advanced liveness detection technique identifies the individual being scanned as alive or fake in biometric security systems. It has all the potential to enhance security, reliability, and effectiveness of a biometric system and protect against unauthorized access. Today's technological advances can be exploited by fraudsters and hence, liveness detection is a vital feature for modern biometric-based recognition solutions.

Also with the advent of latest generation powerful processors capable of complex high speed computing the ability to merge more than one type of algorithm algorithms, we would be able to setup a fool-proof biometric security or spoof-proof biometric security system in this situation.

CONCLUSION

This project is aimed to build a face recognition system that we can implement in a numerous way. We will be using OpenCV to detect faces and eyes and then we intend to use a CNN model to differentiate the fake/spoofed face against a real face.

Detection techniques are the effects of illumination change, effects of amplified noise on images which damages the texture information. For blinking and movement of eyes based detection methods, eyes glasses which causes reflection must be considered for future development of liveness detection solutions. Furthermore, Non-interactive video sequences must include interactive sequences where the users perform certain tasks.

REFERENCES

[1] Haonan Chen, Yaowu Chen, Xiang Tian, Rongxin Jiang Institute of Advanced Digital Technology and Instrument, Zhejiang University, Zhejiang 310027, China, "A Cascade Face Spoofing Detector Based on Face Anti-Spoofing R-CNN and Improved Retinex LBP", IEEE,November 25, 2019.

[2] M. KÕllÕolu, M. TaúkÕran, N. Kahraman, Yildiz Technical University, Istanbul, Turkey, IEEE 15th International Symposium on Applied Machine Intelligence and Informatics, Herl'any, Slovakia, "Anti-Spoofing In Face Recognition with Liveness Detection Using Pupil Tracking", January 28, 2017.

[3]De Freitas Pereira.T, Anjos, A, De Martino.J.M, Marcel.S, "International Conference on Biometris IEEE paper on Face Anti-spoofing Countermeasures in Real World Scenario", 2015.

[4]Theja J Jayan, Aneesh R P (Department of ECE, Sree Chitra Thirunal College of Engineering, Thiruvananthapuram, India, International CET Conference on Control, Communication, and Computing (IC4)."Image Quality Measures Based Face Spoofing Detection Algorithm for Online Social Media", 2018.

[5]Tiago de Freitas Pereira, JukkaKomulainen, André Anjos, José Mario De Martino, AbdenourHadid, MattiPietikäinen, Sébastien Marcel EURASIP Journal on "Image and Video Processing", 2014.

[6] Saptarshi Chakraborty1 and Dhrubajyoti Das, Dept. of Computer Science and Engineering, National Institute of Technology, Silchar, India, AN OVERVIEW OF FACE LIVENESS DETECTION, International Journal on Information Theory (IJIT), Vol.3, No.2, April 2014