INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 930-942

e-ISSN : 2583-1062

Impact Factor: 5.725

DETECTION OF FACE MORPHING USING DEEP LEARNING

Mr. A. Ramesh[1], Bheema Sri Lakshmi[2], Dasari Narendar[3], Moinuddin Mohammed Najeeb[4], Vudaru Sai[5]

[1]Associate. Professor, CSE Dept, ACE Engineering College, Hyderabad, India.

[2,3,4,5]Student, CSE Dept, ACE Engineering College, Hyderabad, India.

ABSTRACT

Identification by biometric features has become more popular in the last decade. High quality video and fingerprint sensors have become less expensive and are nowadays standard components in many mobile devices. Thus, many devices can be unlocked via fingerprint or face verification. The state-of-the-art accuracy of biometric facial recognition systems prompted even systems that need high security standards like border control at airports to rely on biometric systems. While most biometric facial recognition systems perform quite accurate under a controlled environment, they can easily be tricked by morphing attacks. The concept of a morphing attack is to create a synthetic face image that contains characteristics of two different individuals and to use this image on a document or as reference image in a database. Using this image for authentication, a biometric facial recognition system accepts both individuals. In this paper, we propose a morphing attack detection approach based on convolutional neural networks.

We present an automatic morphing pipeline to generate morphing attacks, train neural networks based on this data and analyze their accuracy. The accuracy of different well-known network architectures is compared and the advantage of using pretrained networks compared to networks learned from scratch is studied.

1. INTRODUCTION

Biometric facial recognition systems are nowadays present in many areas of daily life. They are used to identify people, find pictures of the same person in your digital image collection, to make suggestions for tagging people in social media or for verification tasks like unlocking a phone or a computer Apart from these consumer market applications, biometric facial recognition systems found also their way into sovereign tasks like automatic border control at airports. In particular, for these tasks the verification system has to be reliable and secure. Even though biometric facial recognition systems achieve false rejection rates below 1% at a false acceptance rate of 0.1% in a controlled environment [1], they can easily be tricked by a specific attack, known as morphing attack [2]. The concept of this attack is to create a synthetic face image that is, in the eye of the recognition system, similar to the faces of two different individuals.

Thus, both can use the same synthetic face image for authentication. This attack is usually performed by creating a face image that contains characteristics of both faces, for example by face morphing. Since such an attack has a drastic impact on the authenticity of its underlying system, its automatic detection is of outmost importance. The detection of attacks by image manipulation has been studied for various tasks. Several publications deal with the detection of resampling artifacts [3, 4], the use of specific filters, e.g. median filters [5], or JPEG-double compression [6, 7]. Beside these analyses based on signal theory, several authors proposed image forgery detection methods based on semantic image content by analyzing reflections and shadows [8, 9]. Recently, the detection and analysis of forged face images with the purpose of tricking a facial recognition system became interesting to many researchers, which were certainly influenced by the publication of a manual for morphing attacks from Ferrara et al. [2]. Markeshia et al. [10] proposed an approach for automatic generation of facial morphs and their detection based on the distribution of Benford features extracted from quantized DCT coefficients of JPEG-compressed morphs. Raghavendra et al. [11] presented a morphing detection method based on binary statistical images features and evaluated its accuracy on manually created morphed face images.

2. PROBLEM STATEMENT

Face recognition systems (FRSs), integral to real-time applications such as border control, face a significant vulnerability in the form of face morphing attacks. Malicious actors can exploit this weakness by generating morphed face images to obtain electronic machinereadable travel documents (eMRTDs) or e-passports. This allows them to cross borders undetected, presenting a serious threat to security. The survey aims to systematically review the progress in face morphing attacks, covering morph generation and detection techniques. It addresses the urgent need for effective morph attack detection (MAD) algorithms, emphasizing a stringent taxonomy and the importance of standardized benchmarks for evaluation. By exploring competitions, vulnerability assessments, and performance metrics, the survey provides a comprehensive understanding of the current landscape. The identified open challenges underscore the ongoing necessity for research and development to ensure the resilience of biometric systems in the face of evolving threats.

2.1 EXISTING SYSTEM

The development and implementation of Morph Attack Detection (MAD) algorithms. These algorithms aim to detect morphed face images used for bypassing Face Recognition Systems (FRSs) in automatic border control gates. The system includes techniques for generating morphed face images as well as state-of-the-art MAD algorithms based on a comprehensive taxonomy. Additionally, the system provides access to public databases for benchmarking new MAD algorithms in a reproducible manner. The system also includes vulnerability assessments, performance evaluation metrics, outcomes of competitions, and benchmarking results. It addresses the challenges in face morphing attacks and aims to improve the security of biometric systems.

2.2 PROPOSED SYSTEM

The proposed system aims to enhance the security of face recognition systems (FRSs) by developing advanced techniques for detecting face morphing attacks. The system will utilize state-of-the-art deep learning algorithms, such as convolutional neural networks (CNNs) and generative adversarial networks (GANs), to improve the accuracy of morphed face image detection. Additionally, the proposed system will include a comprehensive database of morphed face images for training and testing purposes. To ensure the effectiveness of the system, it will also incorporate real-time monitoring capabilities and will be designed to integrate seamlessly with existing FRSs, particularly in critical applications such as border control. Furthermore, the system will provide detailed vulnerability assessments, performance evaluation metrics, and regular updates to address emerging threats in biometric security.

3. LITERATURE SURVEY

Title: -Morphingattack potential

Description: - In security systems the risk assessment in the sense of commoncriteria testing is a very relevant topic; this requires quantifying the attack potential in terms of the expertise of the attacker, his knowledge about the target and access to equipment. Contrary to those attacks, the recently revealed morphing attacks against Face Recognition Systems (FRSs) cannot be assessed by any of the abovecriteria. But not all morphing techniques pose the same risk for an operational face recognition system. This paper introduces with the Morphing Attack Potential (MAP) a consistent methodology, that can quantify the risk, which a certain morphing attack creates.

Disadvantages: - The model introduces the Morphing Attack Potential (MAP) as a methodology to quantify the risk posed by morphing attacks against Face Recognition Systems (FRSs). While it addresses the limitations oftraditional risk assessment criteria inevaluating morphing attacks, the abstract has some notable disadvantages. Firstly, it lacks specific details about how MAP is calculated or what factors it considers in quantifying the risk. Additionally, there is no mention ofthe methodology's reliability or validation. The abstract could benefit from providing more information on the practical implementation of MAP, its applicability across different morphing techniques, and any experimental results demonstrating its effectiveness. Furthermore, clarity on how MAP contributes to addressing the specific challenges posed by morphing attacks would enhance the abstract's comprehensibility.

Title: -On resampling detection in re-compressed images

Description: - Resampling detection has become astandard tool in digital image forensics. This paper investigates the important case of resampling detection in re-compressed JPEG images. We show how blocking artifacts of the previous compression step can help to increase the otherwise drastically reduced detection performance in JPEG compressed images. We givea formulation on how affine transformations of JPEG compressed images affect state-of-the-art resampling detectors and derive a new efficient detection variant, which better suits this relevant detection scenario. The principal appropriateness of using JPEG pre-compression artifacts forthe detection of resampling in re-compressed images is backed with experimental evidence on a large image set and for a variety of different JPEG qualities.

Disadvantages: The model provides insights into the investigation of resampling detection in re-compressed JPEG images, with a focus on utilizing blocking artifacts from the previous compression step to improve detection performance. However, it has some notable disadvantages.

Firstly, it lacks clarity in explaining the specific mechanisms by which blocking artifacts aid in detection improvement. The abstract could benefit from a more detailed explanation of the proposed formulation and how affine transformations in JPEG compressed images impact state-of-the-artresampling detectors.

Title: -Exposing digital forgeries through specular highlightson the eye

Description: - When creating a digital composite of two people, it is difficult to exactly match the lighting conditions under which each individual was originally photographed. In many situations, thelight source in a scene gives rise to a specular highlight on the eyes. We show how the direction to a light source can be estimated from this highlight. Inconsistencies in lighting across an image are then used to reveal traces of digital tampering.

Disadvantages: - The model discusses the challengesof precisely matching lighting conditions in digital composites of two people and proposes a method to estimate the direction to a light source using specular highlights onthe eyes. While the idea is interesting, the abstract has several disadvantages. Firstly, it lacks specificity in describing the proposed method; details on the techniques used for estimating light direction and identifying inconsistencies in lighting are not provided. This absence of technical details hinders the reader's understanding of the novelty and effectiveness of the approach.

Title: -Analyzing classifiers: Fisher vectors anddeep neuralnetworks

Description: - Fisher vector (FV) classifiers and Deep Neural Networks (DNNs) arepopular and successful algorithms for solving image classification problems. However, both are generally considered black box ‘predictors as the non-linear transformations involved have so far prevented transparent and interpretable reasoning. Recently, aprincipled technique, Layer-wise Relevance Propagation (LRP), has been developed in order to better comprehend the inherent structuredreasoning of complex nonlinear classification models such as Bag of Feature models or DNNs. In thispaper we (1) extend the LRP framework also for Fisher vector classifiers and then use it as analysis tool to (2) quantify the importance of context for classification,(3) qualitatively compare DNNs against FV classifiers in terms of important image regions and (4) detect potential flaws and biases in data.All experiments are performed onthe PASCAL VOC 2007 and ILSVRC 2012 data sets.

Disadvantages: - The model mentions experiments on specific datasets (PASCAL VOC 2007 and ILSVRC 2012), but it doesnot provide any results or findings from these experiments, leaving the reader without a sense of the outcomes.

Title: -A multi- purpose image counter- anti-forensicmethod using convolution al neural networks

Description: - During the past decade, image forensics has made rapid progress due to the growing concern of image content authenticity. In orderto remove or conceal the traces thatforensics based on, some farsightedforgers take advantage of so-called anti-forensics to make their forgerymore convincing. To rebuild the credibility of forensics, many countermeasures against anti- forensics have been proposed. This paper presents a multi-purpose approach to detect various anti- forensics based on the architecture of Convolutional Neural Networks (CNN), which can automatically extract features and identify the forged types. Our model can detect various image anti-forensics both in binary and multi-class decision effectively. Experimental results show that the proposed method performs well for multiple well- known image anti-forensic methods.

Disadvantages: - The model does not address the broader context of ethicalconsiderations related to image forensics, such as potential misuse or privacy concerns. A brief discussion on the ethical implications of developing anti-forensic detection methods would adddepth to the abstract.

Title: -Video basedfacial reanimation

Description: - Generating photorealistic facial animations is still a challenging task in computer

4. OBJECTIVE OF THE PROJECT

Knowledge Dissemination:

Disseminate knowledge about the potential risks and vulnerabilities faced by FRSs, especially concerning the use of face morphing attacks to manipulate electronic travel documents.

Survey of Face Morphing Landscape:

Present a systematic survey of the current state of face morphing attacks, covering both the generation of morphed face images and the development of detection algorithms.

Taxonomy and Classification:

Introduce a stringent taxonomy for the classification of morph attack detection (MAD) algorithms, providing a structured framework for understanding different techniques and approaches.

Benchmarking and Evaluation:

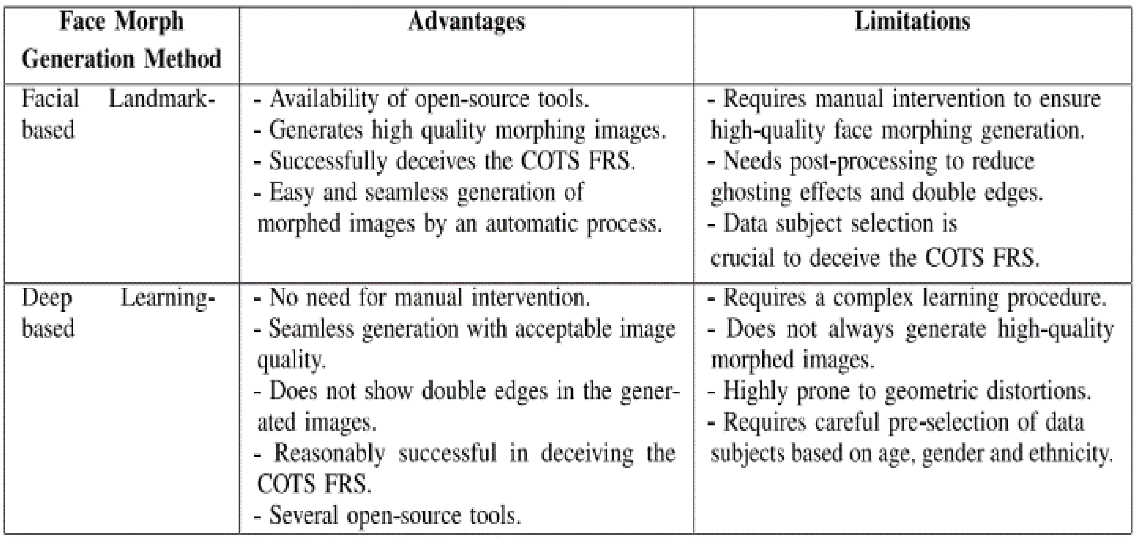
Emphasize the importance of standardized benchmarks and public databases for evaluating the effectiveness of MAD algorithms, including insights from competitions, vulnerability assessments, and performance metrics.

Identification of Challenges:

Highlight open challenges in the field, recognizing areas that require further attention and research to enhance the security of biometric systems against face morphing attacks

When the majority of nodes agree (aka gain consensus), the system updates itself accordingly (ie. adopting the new information into all copies of the ledger, or rejecting it). Distributed ledgers run without a central authority, revolutionizing the way we think of democracy.

5. ADVANTAGES AND LIMITATIONS



Face Morph Generation Method	Advantages	Limitations
Facial Landmark-based	- Availability of open-source tools. - Generates high quality morphing images. - Successfully deceives the COTS FRS. - Easy and seamless generation of morphed images by an automatic process.	- Requires manual intervention to ensure high-quality face morphing generation. - Needs post-processing to reduce ghosting effects and double edges. - Data subject selection is crucial to deceive the COTS FRS.
Deep Learning-based	- No need for manual intervention. - Seamless generation with acceptable image quality. - Does not show double edges in the generated images. - Reasonably successful in deceiving the COTS FRS. - Several open-source tools.	- Requires a complex learning procedure. - Does not always generate high-quality morphed images. - Highly prone to geometric distortions. - Requires careful pre-selection of data subjects based on age, gender and ethnicity.

6. SYSTEM DESIGN

6.1 UML DIAGRAMS

Class Diagram

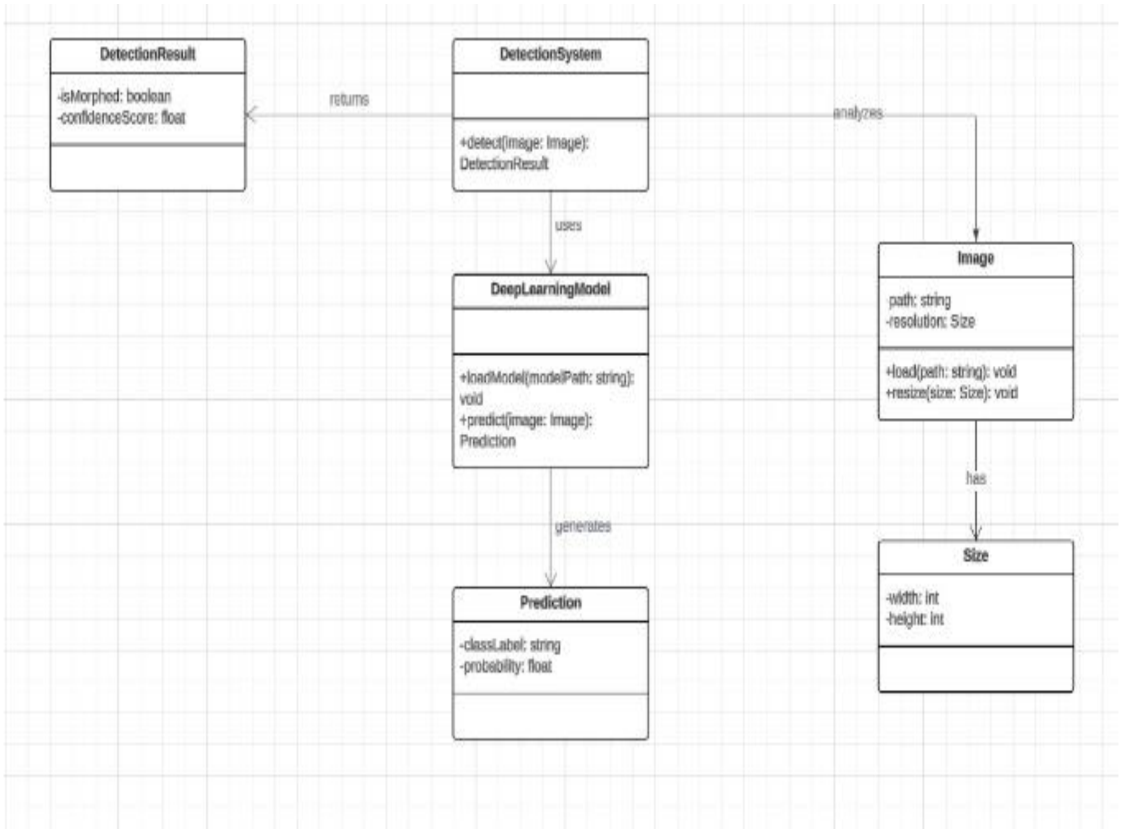


Figure 4 Class Diagram

6.2 USECASE DIAGRAM

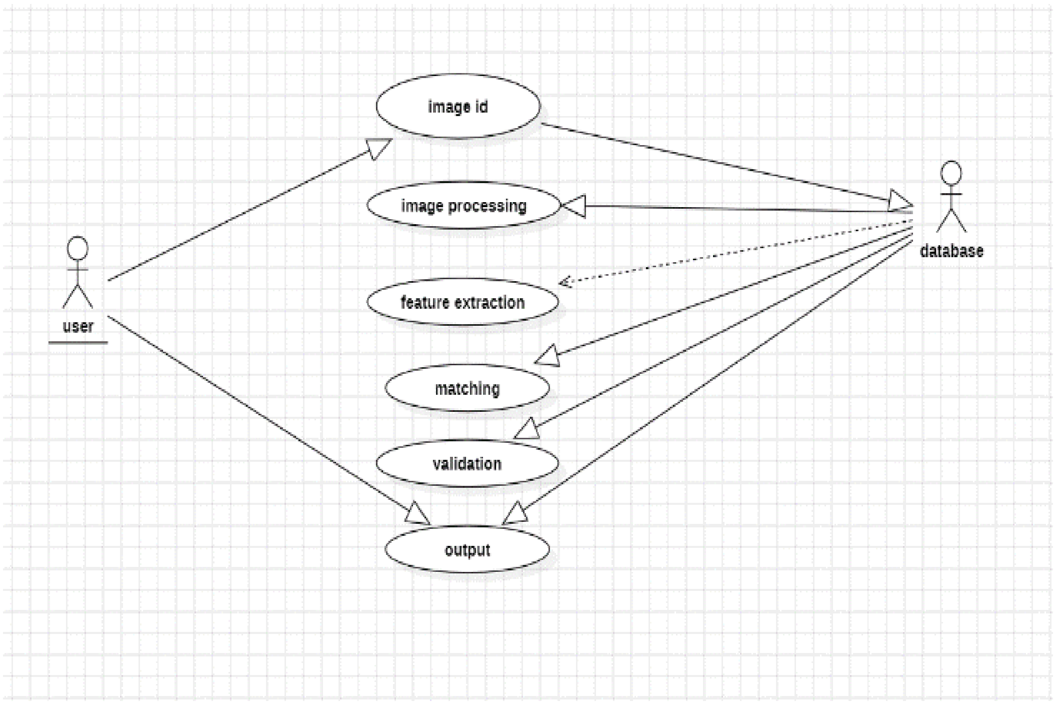


Figure5:Usecase Diagram

6.3 COMPONENT DIAGRAM

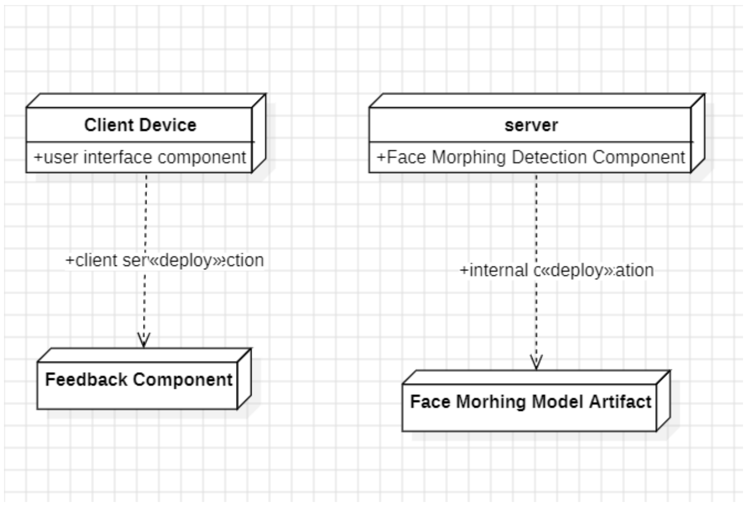

Client Device
+user interface component
server
+Face Morphing Detection Component
+client ser«deploy»ction
+internal c«deploy»ation
Feedback Component
Face Morhing Model Artifact

Figure6:Component Diagram

6.4 ACTIVITY DIAGRAM

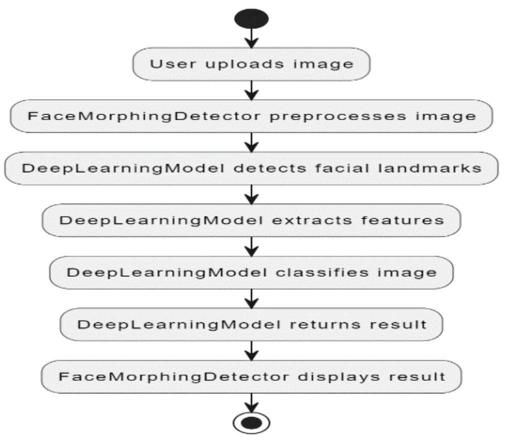

User uploads image
FaceMorphingDetector preprocesses image
DeepLearningModel detects facial landmarks
DeepLearningModel extracts features
DeepLearningModel classifies image
DeepLearningModel returns result
FaceMorphingDetector displays result

Figure.7

7. SOURCE CODE

Face_landmark_Detection

```
import sys
import os
import dlib
import glob
import numpy as np
from skimage import io
import cv2
from imutils import face_utils
class NoFaceFound(Exception):
"""Raised when there is no face found"""
pass
def calculate_margin_help(img1,img2):
size1 = img1.shape
size2 = img2.shape
diff0 = abs(size1[0]-size2[0])//2
diff1 = abs(size1[1]-size2[1])//2
avg0 = (size1[0]+size2[0])//2
avg1 = (size1[1]+size2[1])//2
return [size1,size2,diff0,diff1,avg0,avg1]
def crop_image(img1,img2):
[size1,size2,diff0,diff1,avg0,avg1] = calculate_margin_help(img1,img2)
if(size1[0] == size2[0] and size1[1] == size2[1]):
return [img1,img2]
elif(size1[0] <= size2[0] and size1[1] <= size2[1]):
scale0 = size1[0]/size2[0]
scale1 = size1[1]/size2[1]
if(scale0 > scale1):
res = cv2.resize(img2,None,fx=scale0,fy=scale0,interpolation=cv2.INTER_AREA)
else:
res = cv2.resize(img2,None,fx=scale1,fy=scale1,interpolation=cv2.INTER_AREA)
return crop_image_help(img1,res)
elif(size1[0] >= size2[0] and size1[1] >= size2[1]):
scale0 = size2[0]/size1[0]
scale1 = size2[1]/size1[1]
if(scale0 > scale1):
res = cv2.resize(img1,None,fx=scale0,fy=scale0,interpolation=cv2.INTER_AREA)
else:
res = cv2.resize(img1,None,fx=scale1,fy=scale1,interpolation=cv2.INTER_AREA)
return crop_image_help(res,img2)
elif(size1[0] >= size2[0] and size1[1] <= size2[1]):
return [img1[diff0:avg0,:],img2[:,-diff1:avg1]]
else:
return [img1[:,diff1:avg1],img2[-diff0:avg0,:]]
```

```
def crop_image_help(img1,img2):
[size1,size2,diff0,diff1,avg0,avg1] = calculate_margin_help(img1,img2)
if(size1[0] == size2[0] and size1[1] == size2[1]):
return [img1,img2]
elif(size1[0] <= size2[0] and size1[1] <= size2[1]):
return [img1,img2[-diff0:avg0,-diff1:avg1]]
elif(size1[0] >= size2[0] and size1[1] >= size2[1]):
return [img1[diff0:avg0,diff1:avg1],img2]
elif(size1[0] >= size2[0] and size1[1] <= size2[1]):
return [img1[diff0:avg0,:],img2[:,-diff1:avg1]]
else:
return [img1[:,diff1:avg1],img2[diff0:avg0,:]]
def generate_face_correspondences(theImage1, theImage2):
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('code/utils/shape_predictor_68_face_landmarks.dat')
corresp = np.zeros((68,2))
imgList = crop_image(theImage1,theImage2)
list1 = []
list2 = []
j = 1
for img in imgList:
size = (img.shape[0],img.shape[1])
if(j == 1):
currList = list1
else:
currList = list2
dets = detector(img, 1)
try:
if len(dets) == 0:
raise NoFaceFound
except NoFaceFound:
print("Sorry, but I couldn't find a face in the image.")
j=j+1
for k, rect in enumerate(dets):
shape = predictor(img, rect)
for i in range(0,68):
x = shape.part(i).x
y = shape.part(i).y
currList.append((x, y))
corresp[i][0] += x
corresp[i][1] += y
currList.append((1,1))
currList.append((size[1]-1,1))
currList.append(((size[1]-1)//2,1))
currList.append((1,size[0]-1))
currList.append((1,(size[0]-1)//2))
currList.append(((size[1]-1)//2,size[0]-1))
currList.append((size[1]-1,size[0]-1))
currList.append(((size[1]-1),(size[0]-1)//2))
narray = corresp/2
narray = np.append(narray,[[1,1]],axis=0)
narray = np.append(narray,[[size[1]-1,1]],axis=0)
narray = np.append(narray,[[(size[1]-1)//2,1]],axis=0)
narray = np.append(narray,[[1,size[0]-1]],axis=0)
narray = np.append(narray,[[1,(size[0]-1)//2]],axis=0)
narray = np.append(narray,[[(size[1]-1)//2,size[0]-1]],axis=0)
narray = np.append(narray,[[size[1]-1,size[0]-1]],axis=0)
narray = np.append(narray,[[(size[1]-1),(size[0]-1)//2]],axis=0)
return [size,imgList[0],imgList[1],list1,list2,narray]
```

Face_landmark_Detection

```
import numpy as np
import cv2
import sys
import os
import math
from subprocess import Popen, PIPE
from PIL import Image
def apply_affine_transform(src, srcTri, dstTri, size) :
warpMat = cv2.getAffineTransform(np.float32(srcTri), np.float32(dstTri))
dst = cv2.warpAffine(src, warpMat, (size[0], size[1]), None, flags=cv2.INTER_LINEAR, borderMode=cv2.BORDER_REFLECT_101)
return dst
def morph_triangle(img1, img2, img, t1, t2, t, alpha) :
r1 = cv2.boundingRect(np.float32([t1]))
r2 = cv2.boundingRect(np.float32([t2]))
r = cv2.boundingRect(np.float32([t]))
t1Rect = []
t2Rect = []
tRect = []
for i in range(0, 3):
tRect.append(((t[i][0] - r[0]),(t[i][1] - r[1])))
t1Rect.append(((t1[i][0] - r1[0]),(t1[i][1] - r1[1])))
t2Rect.append(((t2[i][0] - r2[0]),(t2[i][1] - r2[1])))
mask = np.zeros((r[3], r[2], 3), dtype = np.float32)
cv2.fillConvexPoly(mask, np.int32(tRect), (1.0, 1.0, 1.0), 16, 0)
img1Rect = img1[r1[1]:r1[1] + r1[3], r1[0]:r1[0] + r1[2]]
img2Rect = img2[r2[1]:r2[1] + r2[3], r2[0]:r2[0] + r2[2]]
size = (r[2], r[3])
warpImage1 = apply_affine_transform(img1Rect, t1Rect, tRect, size)
warpImage2 = apply_affine_transform(img2Rect, t2Rect, tRect, size)
imgRect = (1.0 - alpha) * warpImage1 + alpha * warpImage2
img[r[1]:r[1]+r[3], r[0]:r[0]+r[2]] = img[r[1]:r[1]+r[3], r[0]:r[0]+r[2]] * ( 1 - mask ) + imgRect * mask
def generate_morph_sequence(duration,frame_rate,img1,img2,points1,points2,tri_list,size,output):
num_images = int(duration*frame_rate)
```

```
p = Popen(['ffmpeg', '-y', '-f', 'image2pipe', '-r', str(frame_rate),'-s',str(size[1])+'x'+str(size[0]), '-i', '-', '-c:v', 'libx264', '-crf', '25','-vf','scale=trunc(iw/2)*2:trunc(ih/2)*2','-pix_fmt','yuv420p', output], stdin=PIPE)
for j in range(0, num_images):
img1 = np.float32(img1)
img2 = np.float32(img2)
points = []
alpha = j/(num_images-1)
for i in range(0, len(points1)):
x = (1 - alpha) * points1[i][0] + alpha * points2[i][0]
y = (1 - alpha) * points1[i][1] + alpha * points2[i][1]
points.append((x,y))
morphed_frame = np.zeros(img1.shape, dtype = img1.dtype)
for i in range(len(tri_list)):
x = int(tri_list[i][0])
y = int(tri_list[i][1])
z = int(tri_list[i][2])
t1 = [points1[x], points1[y], points1[z]]
t2 = [points2[x], points2[y], points2[z]]
t = [points[x], points[y], points[z]]
morph_triangle(img1, img2, morphed_frame, t1, t2, t, alpha)
pt1 = (int(t[0][0]), int(t[0][1]))
pt2 = (int(t[1][0]), int(t[1][1]))
pt3 = (int(t[2][0]), int(t[2][1]))
cv2.line(morphed_frame, pt1, pt2, (255, 255, 255), 1, 8, 0)
cv2.line(morphed_frame, pt2, pt3, (255, 255, 255), 1, 8, 0)
cv2.line(morphed_frame, pt3, pt1, (255, 255, 255), 1, 8, 0)
res = Image.fromarray(cv2.cvtColor(np.uint8(morphed_frame), cv2.COLOR_BGR2RGB))
res.save(p.stdin,'JPEG')
p.stdin.close()
p.wait()
```

Face alignment

```
import numpy as np
import scipy.ndimage
import os
import PIL.Image
def image_align(src_file, dst_file, face_landmarks, output_size=1024, transform_size=4096, enable_padding=True, x_scale=1, y_scale=1, em_scale=0.1, alpha=False):
# Align function from FFHQ dataset pre-processing step
# https://github.com/NVlabs/ffhq-dataset/blob/master/download_ffhq.py
lm = np.array(face_landmarks)
lm_chin          = lm[0  : 17]  # left-right
lm_eyebrow_left  = lm[17 : 22]  # left-right
lm_eyebrow_right = lm[22 : 27]  # left-right
lm_nose          = lm[27 : 31]  # top-down
lm_nostrils      = lm[31 : 36]  # top-down
lm_eye_left      = lm[36 : 42]  # left-clockwise
lm_eye_right     = lm[42 : 48]  # left-clockwise
```

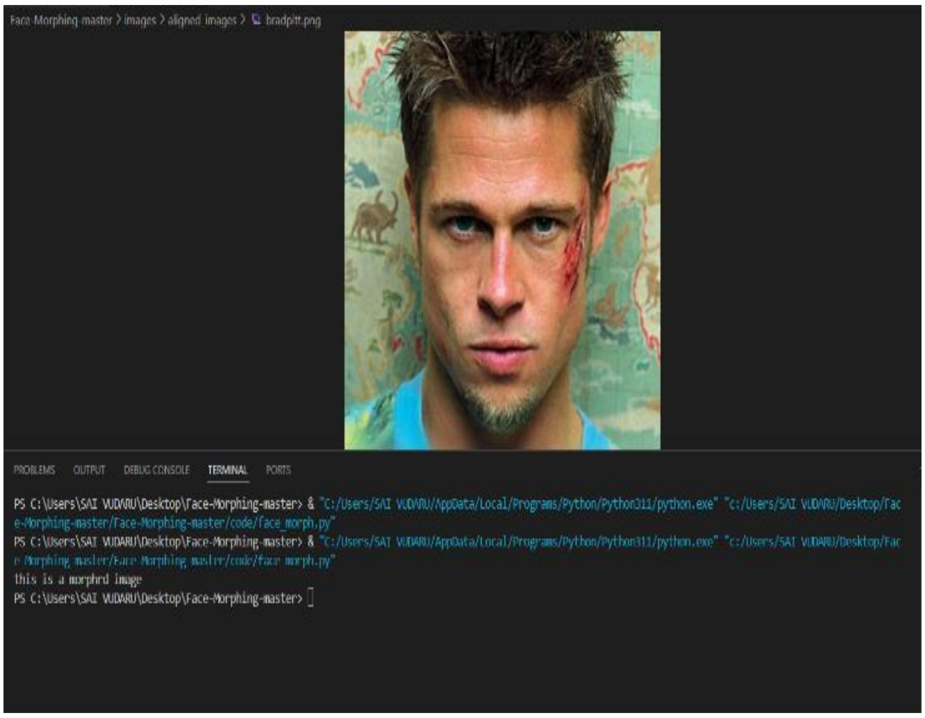
```
lm_mouth_outer  = lm[48 : 60]  # left-clockwise
lm_mouth_inner  = lm[60 : 68]  # left-clockwise
# Calculate auxiliary vectors.
eye_left     = np.mean(lm_eye_left, axis=0)
eye_right    = np.mean(lm_eye_right, axis=0)
eye_avg      = (eye_left + eye_right) * 0.5
eye_to_eye   = eye_right - eye_left
mouth_left   = lm_mouth_outer[0]
mouth_right  = lm_mouth_outer[6]
mouth_avg    = (mouth_left + mouth_right) * 0.5
eye_to_mouth = mouth_avg - eye_avg
# Choose oriented crop rectangle.
x = eye_to_eye - np.flipud(eye_to_mouth) * [-1, 1]
x /= np.hypot(*x)
x *= max(np.hypot(*eye_to_eye) * 2.0, np.hypot(*eye_to_mouth) * 1.8)
x *= x_scale
y = np.flipud(x) * [-y_scale, y_scale]
c = eye_avg + eye_to_mouth * em_scale
quad = np.stack([c - x - y, c - x + y, c + x + y, c + x - y])
qsize = np.hypot(*x) * 2
# Load in-the-wild image.
if not os.path.isfile(src_file):
print('\nCannot find source image. Please run "--wilds" before "--align".')
return
img = PIL.Image.open(src_file).convert('RGBA').convert('RGB')
# Shrink.
shrink = int(np.floor(qsize / output_size * 0.5))
if shrink > 1:
rsize = (int(np.rint(float(img.size[0]) / shrink)), int(np.rint(float(img.size[1]) / shrink)))
img = img.resize(rsize, PIL.Image.ANTIALIAS)
quad /= shrink
qsize /= shrink
# Crop.
border = max(int(np.rint(qsize * 0.1)), 3)
crop = (int(np.floor(min(quad[:,0]))), int(np.floor(min(quad[:,1]))), int(np.ceil(max(quad[:,0]))),
int(np.ceil(max(quad[:,1]))))
crop = (max(crop[0] - border, 0), max(crop[1] - border, 0), min(crop[2] + border, img.size[0]), min(crop[3] + border,
img.size[1]))
if crop[2] - crop[0] < img.size[0] or crop[3] - crop[1] < img.size[1]:
img = img.crop(crop)
quad -= crop[0:2]
Pad.
pad = (int(np.floor(min(quad[:,0]))), int(np.floor(min(quad[:,1]))), int(np.ceil(max(quad[:,0]))),
int(np.ceil(max(quad[:,1]))))
pad = (max(-pad[0] + border, 0), max(-pad[1] + border, 0), max(pad[2] - img.size[0] + border, 0), max(pad[3] -
img.size[1] + border, 0))
if enable_padding and max(pad) > border - 4:
```

```
pad = np.maximum(pad, int(np.rint(qsize * 0.3)))
img = np.pad(np.float32(img), ((pad[1], pad[3]), (pad[0], pad[2]), (0, 0)), 'reflect')
h, w, _ = img.shape
y, x, _ = np.ogrid[:h, :w, :1]
mask = np.maximum(1.0 - np.minimum(np.float32(x) / pad[0], np.float32(w-1-x) / pad[2]), 1.0 - np.minimum(np.float32(y) / pad[1], np.float32(h-1-y) / pad[3]))
blur = qsize * 0.02
img += (scipy.ndimage.gaussian_filter(img, [blur, blur, 0]) - img) * np.clip(mask * 3.0 + 1.0, 0.0, 1.0)
img += (np.median(img, axis=(0,1)) - img) * np.clip(mask, 0.0, 1.0)
img = np.uint8(np.clip(np.rint(img), 0, 255))
if alpha:
mask = 1-np.clip(3.0 * mask, 0.0, 1.0)
mask = np.uint8(np.clip(np.rint(mask*255), 0, 255))
img = np.concatenate((img, mask), axis=2)
img = PIL.Image.fromarray(img, 'RGBA')
else:
img = PIL.Image.fromarray(img, 'RGB')
quad += pad[:2]
# Transform.
img = img.transform((transform_size, transform_size), PIL.Image.QUAD, (quad + 0.5).flatten(), PIL.Image.BILINEAR)
if output_size < transform_size:
img = img.resize((output_size, output_size), PIL.Image.ANTIALIAS)
# Save aligned image.
img.save(dst_file, 'PNG'
```

8. OUTPUT

OUTPUT WINDOWS:

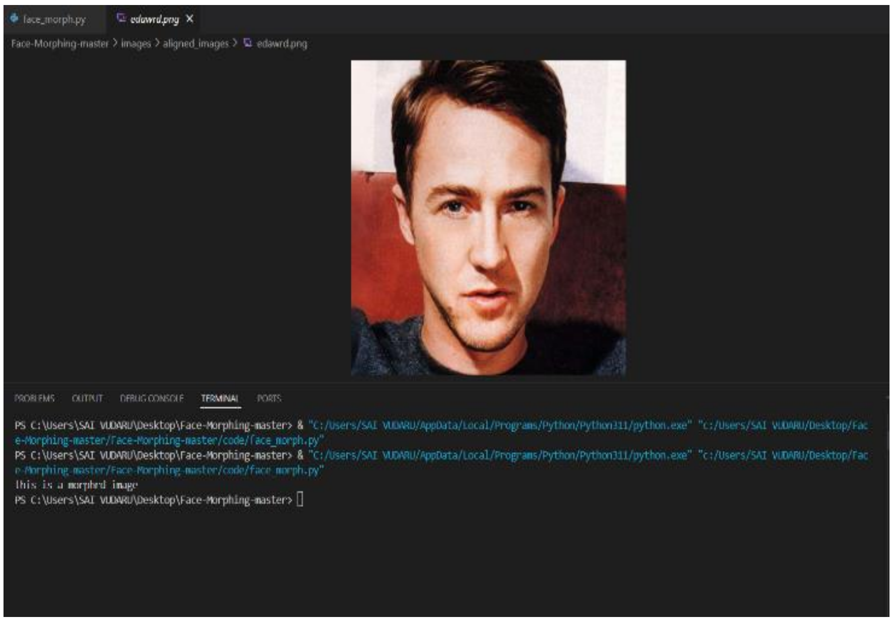
OUTPUT 1


Face-Morphing-master > images > aligned_images > bradpitt.png
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

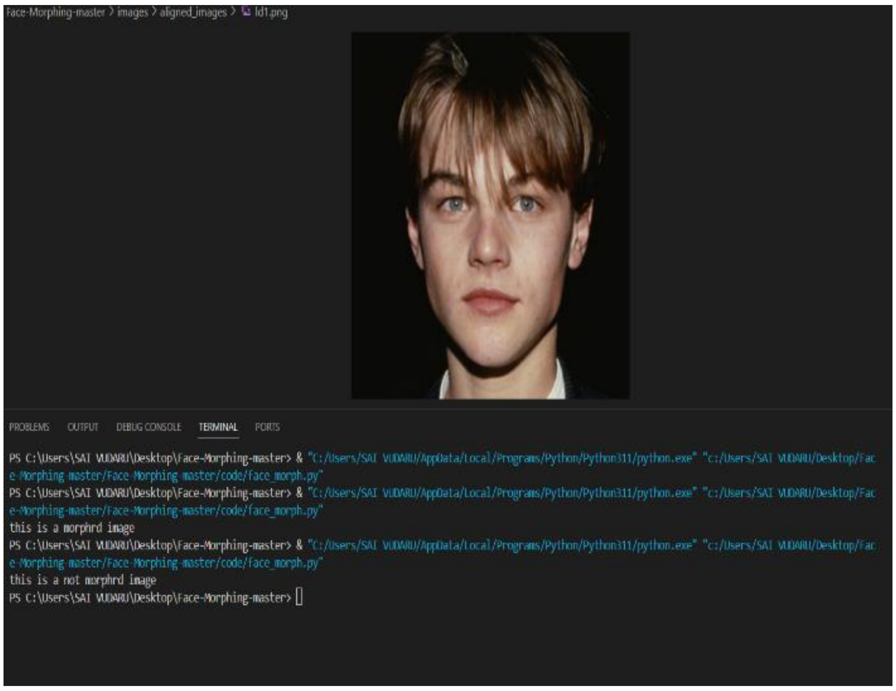
```
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> & "C:/Users/SAI VUDARU/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/SAI VUDARU/Desktop/Face-Morphing-master/Face-Morphing-master/code/face_morph.py"
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> & "C:/Users/SAI VUDARU/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/SAI VUDARU/Desktop/Face-Morphing-master/Face-Morphing-master/code/face_morph.py"
this is a morphrd image
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> 
```

Figure 1:Output WindOW

OUTPUT 2:



```
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> & "C:/Users/SAI VUDARU/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/SAI VUDARU/Desktop/Fac
e-Morphing-master/Face-Morphing-master/code/face_morph.py"
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> & "C:/Users/SAI VUDARU/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/SAI VUDARU/Desktop/Fac
e-Morphing-master/Face-Morphing-master/code/face_morph.py"
this is a morphrd image
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> 
```



```
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> & "C:/Users/SAI VUDARU/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/SAI VUDARU/Desktop/Fac
e-Morphing-master/Face-Morphing-master/code/face_morph.py"
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> & "C:/Users/SAI VUDARU/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/SAI VUDARU/Desktop/Fac
e-Morphing-master/Face-Morphing-master/code/face_morph.py"
this is a morphrd image
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> & "C:/Users/SAI VUDARU/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/SAI VUDARU/Desktop/Fac
e-Morphing-master/Face-Morphing-master/code/face_morph.py"
this is a not morphrd image
PS C:\Users\SAI VUDARU\Desktop\Face-Morphing-master> 
```

9. CONCLUSION

we proposed a morphing attack detection method based on deep convolution neural networks. A fully automatic face image morphing pipeline with exchangeable components was presented that was used to create training and test samples for the networks. Instead of detecting classical traces of tampering, e.g. caused by resampling or JPEG double compression, and their anti-forensic methods we focused on semantic artifacts. To avoid learning a detector for these classical tampering traces, all images were preprocessed by scaling, rotating and cropping, before feeding them to the network. In addition, we added different kinds of noise and blur to the training and test data. We trained three different convolutional neural network architectures from scratch and starting with pretrained networks.

The FRR of our trained networks differ between 3.5% and 16.2% and the FAR between 0.8% and 2.2%. The VGG19 (pretrained) achieved for both rates the best result with a FRR of 3.5% and a FAR of 0.8%. The pretrained networks outperformed the networks trained from scratch for every architecture. This suggests that the features learned for object classification are also useful for detection of morphing attacks. In future work, we plan to analyze the decisions made by our networks. In particular, we plan to study the regions that contribute to the decision of a network and analyze the differences between different architectures and pretrained networks using the LRP toolbox.

10. REFERENCES

[1] Spreeuwers, L.J., Hendrikse, A.J., Gerritsen, K.J.: Evaluation of automatic face recognition for automatic border control on actual data recorded of travellers at schiphol airport. In: Proceedings of the International Conference of Biometrics Special Interest Group (BIOSIG). (Sept 2012) 1–6

[2] Ferrara, M., Franco, A., Maltoni, D.: The magic passport. In: IEEE International Joint Conference on Biometrics. (Sept 2014) 1–7

[3] Popescu, A.C., Farid, H.: Exposing digital forgeries by detecting traces of resampling. IEEE Transactions on Signal Processing 53(2) (Feb 2005) 758–767

[4] Kirchner, M., Gloe, T.: On resampling detection in re-compressed images. In: 2009 First IEEE International Workshop on Information Forensics and Security (WIFS). (Dec 2009) 21–25

[5] Kirchner, M., Fridrich, J.: On detection of median filtering in digital images. Proc. SPIE 7541 (2010) 754110–754110–12

[6] Luk´aˇs, J., Fridrich, J.: Estimation of primary quantization matrix in double compressed jpeg images. In: Proc. of DFRWS. (2003)

[7] Farid, H.: Exposing digital forgeries from jpeg ghosts. IEEE Transactions on Information Forensics and Security 4(1) (March 2009) 154–160

[8] Johnson, M.K., Farid, H.: Exposing digital forgeries through specular highlights on the eye. In: Information Hiding. Volume 4567 of Lecture Notes in Computer Science. (2008) 311–325

[9] Kee, E., O'brien, J.F., Farid, H.: Exposing photo manipulation from shading and shadows. ACM Trans. Graph. 33(5) (September 2014) 165:1–165:21

[10] Makrushin, A., Neubert, T., Dittmann, J.: Automatic generation and detection of visually faultless facial morphs. In: Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017). (2017) 39–50