A Project Report

on

HUMAN_FACE_DETECTION

*carried out as part of the **Mini Project DS3130** Submitted*

by

Vaibhav Nagar (209309072)
Shubhra Mittal (209309088)

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Data Science

MANIPAL UNIVERSITY
JAIPUR

School of Computing and Information Technology
Department of Information Technology

MANIPAL UNIVERSITY JAIPUR
JAIPUR-303007
RAJASTHAN, INDIA

November 2022

CERTIFICATE

Date: 20 November 2022

This is to certify that the minor project titled **Human Face Detection** is a record of the bonafide work done by Vaibhav Nagar (209309072) and Shubhra Mittal (209309088) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Data Science of Manipal University Jaipur, during the academic year 2022-23.

Dr. Shweta Sharma

Project Guide, Department of Information Technology
Assistant Professor (Senior Scale)

Manipal University Jaipur

ABSTRACT

Face recognition is the problem of identifying and verifying people in a photograph by their face. It is a task that is trivially performed by humans, even under varying light and when faces are changed by age or obstructed with accessories and facial hair. Nevertheless, it has remained a challenging computer vision problem for decades until recently.

Deep learning methods can leverage very large datasets of faces and learn rich and compact representations of faces, allowing modern models to first perform as-well and later to outperform the face recognition capabilities of humans.

With this project we have implemented the following under the face detection umbrella: Implement Bounding Box Regression, Facial Landmark Detection, Blur the Face from the Image, Use Real Time Image Bounding Box. These models have a great significance in various industries including Automobile, Security Access Control, Healthcare and so on.

TABLE OF CONTENTS

INTRODUCTION

Overview

Face recognition is the problem of identifying and verifying people in a photograph by their face. It is a task that is trivially performed by humans, even under varying light and when faces are changed by age or obstructed with accessories and facial hair. Nevertheless, it is remained a challenging computer vision problem for decades until recently. Deep learning methods are able to leverage very large datasets of faces and learn rich and compact representations of faces, allowing modern models to first perform as-well and later to outperform the face recognition capabilities of humans.

Motivation

Facial recognition is a biometric form of technology used to identify human faces. By scanning an individual's face, it will confirm their identity. This could mean that an individual can't have multiple driver's licenses, state IDs or could be identified within a law enforcement database. Facial recognition is one of the key components for future intelligent vehicle applications, like determining whether a person is allowed or authorized to operate a vehicle. The challenge for many security technology companies today is to build a facial recognition security application that is fast and accurate.

Applications

The application of the face recognition system has reinforced its roles in Deep Learning as much as it has done in everyday life.

Social Media: The very first place to go when looking at machine applications is social media. This is because it is the place where computers and humans meet. On social media, the face recognition model has been deployed in FaceTune, SnapChat, DeepFace, and even TikTok.

Security and Verification: Another case that the face recognition model is applied is in FaceID verification. Used especially for security and authentication, you'd find its application on your smartphone (popularly the recent models of iPhone), CCTV, and other login and unlocking devices and platforms.

Other Applications: Face recognition systems have been used to identify subjects in photographs, as access control in security systems, and to scan people's faces in a concert. It has witnessed the most formal applications to the most informal. Not to be too oblivious, national security agencies like the FBI have face recognition in their basic toolkit.

Advantages

- Efficient security: It is faster and more convenient compared to other biometric technologies like fingerprints or retina scans. There are also fewer touchpoints in facial recognition compared to entering passwords or PINs. It supports multifactor authentication for additional security verification.

- Improved accuracy: Facial recognition is a more accurate way to identify individuals than simply using a mobile number, email address, mailing address, or IP address. For example, most exchange services, from stocks to cryptos, now rely on facial recognition to protect customers and their assets.
- Easier integration: Face recognition technology is compatible and integrates easily with most security software. For example, smartphones with front-facing cameras have built-in support for facial recognition algorithms or software code.

Problem Statement

Given an image, uniquely identifying a person's face from the source. Using deep learning models and techniques like Bounding Box Regression, Facial Landmark Detection to identify the person. Demonstrating a security application, Blur the Face, of facial recognition. In addition, implementing real-time facial recognition using Real Time Image Bounding Box.

Objectives

We are going to use our model MTCNN for the following processes:

- Implement Bounding Box Regression
- Facial Landmark Detection
- Blur the Face from the Image
- Use Real Time Image Bounding Box

Scope of Project

To create this facial recognition system, we used some off-the-shelf machine learning and computer vision (CV) components. We used the general language choice for machine learning today, Python. TensorFlow, an open-source machine learning and neural network toolkit. TensorFlow is the go-to library for numerical computation and large-scale machine learning. Scikit-learn, simple and efficient tools for data mining and data analysis. Scipy, a free and open source library for scientific and technical computing. Numpy, a Python library supporting large, multi-dimensional arrays with a large library of functions for operating on these arrays. OpenCV, an open-source library of functions aimed at real-time computer vision.

These are all very common tools used for machine learning projects. We've been working with and adapting open source code to tie all of these components together, including the face recognition using Tensorflow.

BACKGROUND DETAILS

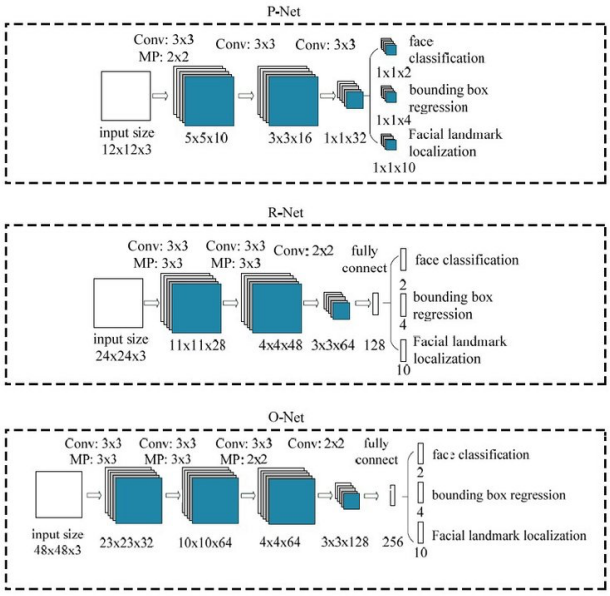
Literature review

One of the most exciting features of artificial intelligence (AI) is undoubtedly face recognition. Research in face recognition started as early as in the 1960s, when early pioneers in the field measured the distances of the various “landmarks” of the face, such as eyes, mouth, and nose, and then computed the various distances in order to determine a person's identity. The work of early researchers was hampered by the limitations of the technology of the day. It wasn't until the late 1980s that we saw the potential of face recognition as a business need. And today, due to the technological advances in computing power, face recognition is gaining popularity and can be performed easily, even from mobile devices.

SYSTEM DESIGN & METHODOLOGY

System Architecture

The model used for this project is MTCNN. The MTCNN is popular because it achieved then state-of-the-art results on a range of benchmark datasets, and because it is capable of also recognizing other facial features such as eyes and mouth, called landmark detection. The network uses a cascade structure with three networks; first the image is rescaled to a range of different sizes (called an image pyramid), then the first model (Proposal Network or P-Net) proposes candidate facial regions, the second model (Refine Network or R-Net) filters the bounding boxes, and the third model (Output Network or O-Net) proposes facial landmarks.



Development Environment

We are going to use our model MTCNN to implement bounding box, regression facial landmark detection, blur the face from the image, use real time image bounding box. All of this is implemented on Google Collab in Python language.

Methodology

The test image we used for this project is given below.

Steps to achieve facial recognition:

Step 1: Install MTCNN from pip.

Step 2: Import the required libraries.

Step 3: Upload our test image onto collab and start using the MTCNN Constructor.

Step 4: Use that to detect the faces. It will show the bounding box of the face and co-ordinates for the Face/Faces in the Image.

Step 5: Bounding Box Implementation: define a function which will draw the given box on the basis of the Co-ordinates we get from the MTCNN Detector.

Step 6: Facial Landmark Detection: draw the given facial landmarks and mark them in the way we want to which are given by the MTCNN Constructor.

Step 7: Use the BLUR Function in Open CV.

Step 8: Real Time Image Bounding: open our own Laptop's Camera using OPEN CV and then putting the real time Images through the MTCNN Constructor and giving us back the bounding BOX on our given Face in front of the Camera in Real Time.

IMPLEMENTATION & RESULT

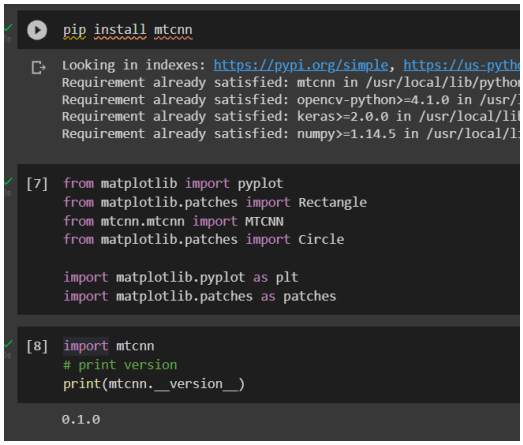
Modules of Implemented Project

Using MTCNN and Matplotlib libraries we implement the following modules.

1. Bounding Box Regression
2. Facial Landmark Detection
3. Blur the Face
4. Real Time Image Bounding Box

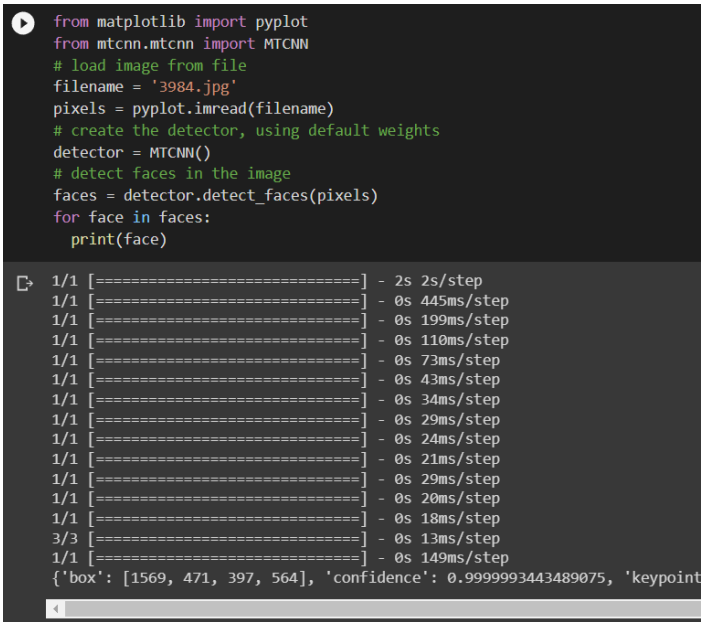
Implementation in detail

First we install MTCNN from pip Second we import all of the libraries involved in the code which would be MATPLOTLIB , MTCNN. We check the version of MTCNN downloaded , we would preferably want it to be the newest version that is 0.1.0.



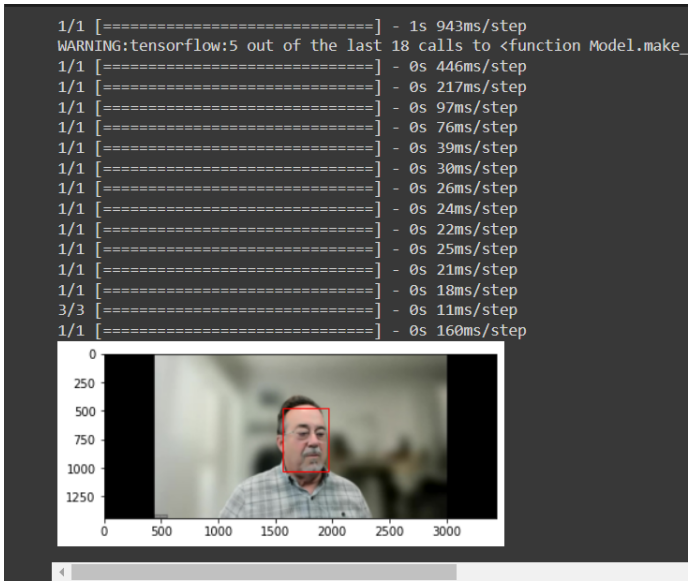
```
pip install mtcnn

Looking in indexes: https://pypi.org/simple, https://us-pyth
Requirement already satisfied: mtcnn in /usr/local/lib/pytho
Requirement already satisfied: opencv-python>=4.1.0 in /usr/
Requirement already satisfied: keras>=2.0.0 in /usr/local/li
Requirement already satisfied: numpy>=1.14.5 in /usr/local/l

[7] from matplotlib import pyplot
    from matplotlib.patches import Rectangle
    from mtcnn.mtcnn import MTCNN
    from matplotlib.patches import Circle

    import matplotlib.pyplot as plt
    import matplotlib.patches as patches

[8] import mtcnn
    # print version
    print(mtcnn.__version__)

    0.1.0
```

After Downloading the necessary entities. We upload our test image onto collab and start using the MTCNN Constructor and use that to detect the faces. It will show the bounding box of the face and co-ordinates for the Face/Faces in the Image.

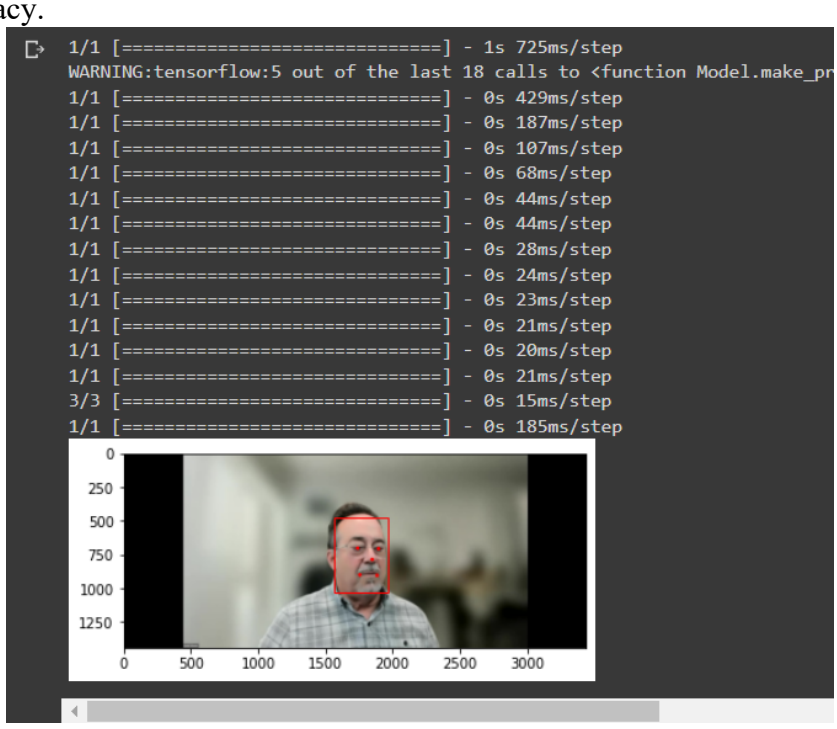


```
from matplotlib import pyplot
from mtcnn.mtcnn import MTCNN
# load image from file
filename = '3984.jpg'
pixels = pyplot.imread(filename)
# create the detector, using default weights
detector = MTCNN()
# detect faces in the image
faces = detector.detect_faces(pixels)
for face in faces:
  print(face)

1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 445ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 73ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 18ms/step
3/3 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 149ms/step
{'box': [1569, 471, 397, 564], 'confidence': 0.9999993443489075, 'keypoint
```

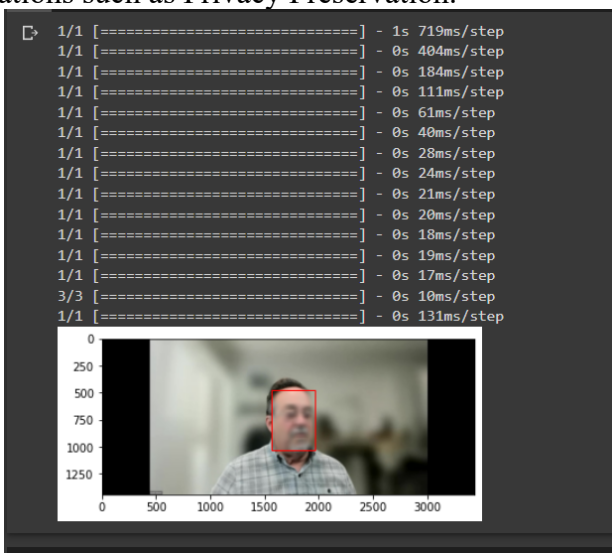
Bounding box: Here we define a function which will draw the given box on the basis of the Co-ordinates we get from the MTCNN Detector. We use this Function on the Filename and the Number of Faces as we can see in the end of the code.



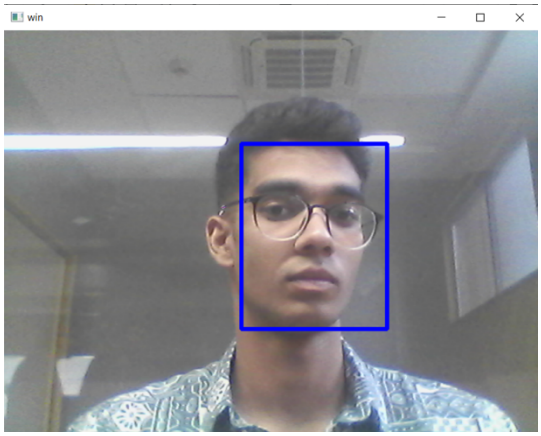
Facial Landmark Detection: In this we draw the given facial landmarks and mark them in the way we want to which are given by the MTCNN Constructor. We draw the given Landmarks which are Eye's , Mouth and Nose. We Draw them with given Red Circles as shown in the Output in the previous Slide. This can be done for multiple faces in a given image and does take a little time but has a good accuracy.



Blurring Face: We Use the BLUR Function in Open CV in the code. We Blur the Face and this may have different used applications such as Privacy Preservation.



Real Time Image Bounding: We use this to open our own Laptop's Camera using OPEN CV and then putting the real time Images through the MTCNN Constructor and giving us back the bounding BOX on our given Face infront of the Camera in Real Time.



Results and Discussion

We successfully implemented facial recognition using Multi-task Cascaded Convolutional Neural Network. We used the model to detect a face from an image source, either already given or in real-time. Furthermore, we demonstrated an application of the same, which is to blur the face (used to protect privacy).

CONCLUSION AND FUTURE PLANS

Conclusion

With this project we gained a deeper insight into deep learning. We worked out the implementation of a special Convolutional Neural Network i.e. Multi-task Cascaded CNN. Multi-task Cascaded Convolutional Networks (MTCNN) is a framework developed as a solution for both face detection and face alignment. The process consists of three stages of convolutional networks that are able to recognize faces and landmark location such as eyes, nose, and mouth. This model can be used for privacy protection (blur the face), security and many more applications.

Future plans

Next steps for this project can be to uniquely identify a person from an image source. Cross-reference a detected face from a given database of identified faces to get information about the person in an image. Further, we can use the above model to create a face detection system that identifies a person in real-time or through video sources.

REFERENCES

Web

[1] Facial Recognition, https://www.codemag.com/Article/2205081/Implementing-Face-Recognition-Using-Deep-Learning-and-Support-Vector-Machines, Accessed on [10-Nov-2022].

[2]MTCNN,https://towardsdatascience.com/face-detection-using-mtcnn-a-guide-for-face-extraction-with-a-focus-on-speed-c6d59f82d49 , Accessed on [15-Nov-2022].