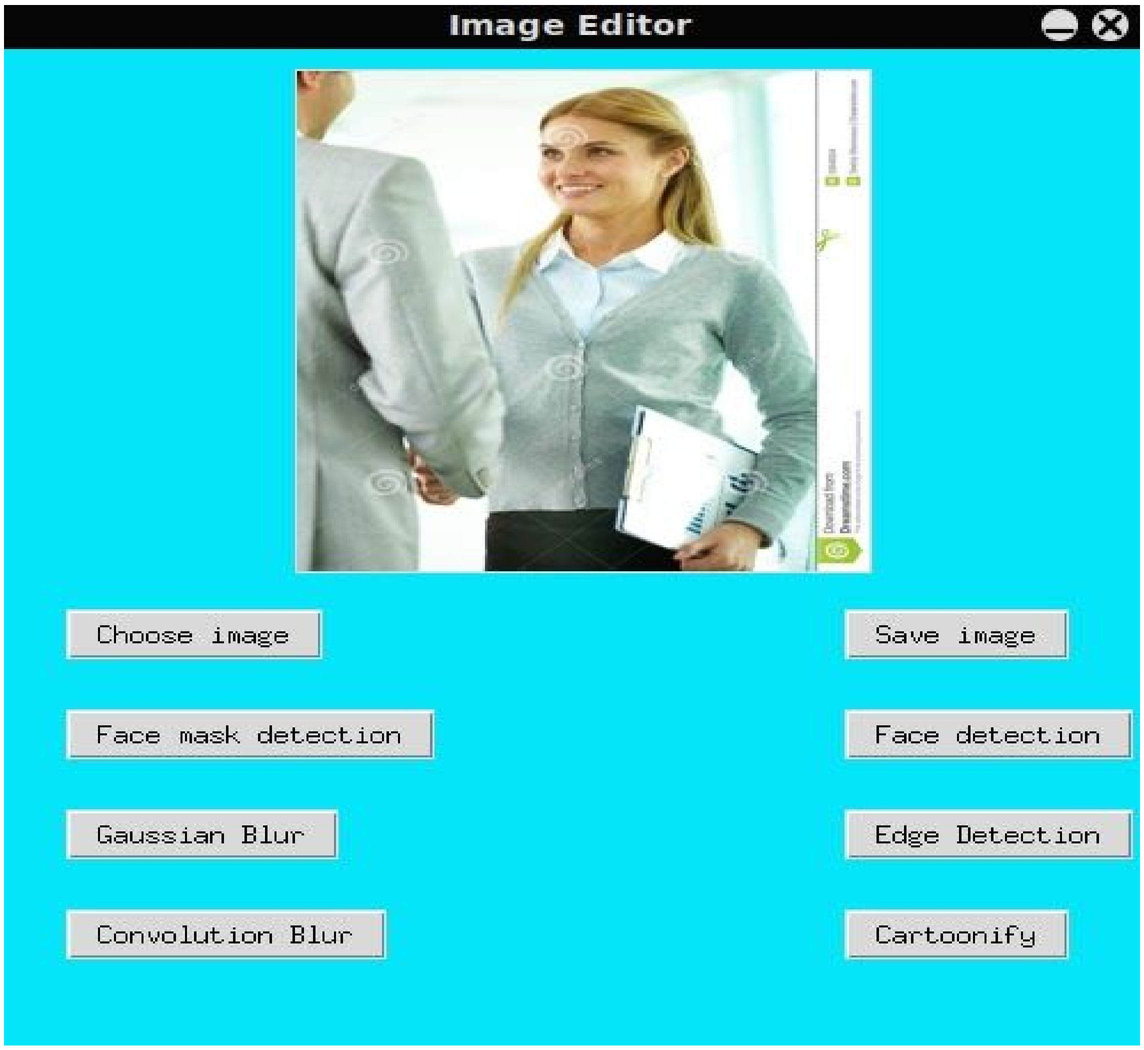

GUI Image Editor

15.06.2020

https://github.com/thelokeshgoel00/photo-filters-python

Lokesh Goel - 11812010 (CS-1)

Hardik Chawla - 11812020 (CS-1)

Overview

This is a GUI based Image Project with a lot of available editing options ranging from simple operations, like blurring of image and cartooning of the image, to some complex operations like face-mask detection, face-detection and Convolution Blur which are implemented through Machine Learning.

Tasks

There are mainly eight different types of tasks(or buttons) available with this editor.

1. Open

We can open an image(only with .jpg or .jpeg extension) through the use of this button. The image can be chosen from any directory. The chosen image is then displayed in front (above the default white screen). All the below operations which you try to apply will all be applied on the image chosen by the user and the result of the operation is displayed.

2. Gaussian Blur

Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function. This operation blurs the image by a value specified by the user. On clicking the button it shows up a dialog box which asks the user to enter a blur value and blurs the image by that value.

3. Convolution Blur

Convolution is a fundamental operation on images in which a mathematical operation is applied to each pixel to get the desired result. This operation blurs the image by a value specified by the user. On clicking the button it shows up a dialog box which asks the user to enter a blur value and blurs the image by that value.

4. Edge Detection

Edges are significant local changes of intensity in a digital image. An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions. Edge Detection is a method of segmenting an image into regions of discontinuity. Thus this method detects all the edges in the image and shows them to the user.

5. Face - mask Detection

This is the best feature of our editor that detects the mask on the face of a person. Its accuracy is around 40% and it is a feature programmed by us in the wake of corona pandemic spreading all over the world. Thus it becomes necessary to wear a mask in this situation so as to decrease the community spread of this virus. Thus this features detects the mask and shows whether it is present or not on the face.

6. Face Detection

This feature detects the face from the image and even it is able to detect the faces of a group of people in an image. This feature encloses the face inside a rectangular box and shows it to the user.

7. Cartooning

This features enables the cartooning of the image chosen by the user and shows the cartooned image back to the user.

8. Save

The last and the important feature of the editor is that it allows the saving of the image to any path chosen by the user. As after applying some operations on the image, a user would like to the save the new image and thus it can be done by just clicking the Save button and specifying the path to save the image.

Model

In This Section, We would like to put light on the Machine Learning Models used for Face-Detection and Face-Mask-Detection

Dataset

We have used a custom dataset provided by Dockship.io for their AI hackathon powered by UnrealAI. The Dataset is custom made with annotations for face available in XML Format. We have provided a training file where this type of dataset could be used. The format of the dataset is similar to PASCAL VOC Dataset and hence can be taken into consideration.

Face Detection

This step is common for both Face Detection and Face Mask Detection. We tried many methods for face detection including traditional Haar Cascades but the accuracy we were looking was not getting achieved. We wanted that side faces, tilted faces could also be discovered. For this same reason, We used MTCNN (Multi-Tasked Cascaded Convolutional Neural Network) model. This is a State-Of-The-Art Model for Face Detection. For details on this model please follow the paper at https://arxiv.org/abs/1604.02878. a

We have used the open-source implementation of this model available at https://github.com/ipazc/mtcnn. This helps us in finding all faces and not just a single face and provide feedback for all the faces. A bounding-box will be generated surrounding the face if face is found successfully.

Face Mask Detection

For the mask Detection, As we have made sure that face is present successfully, We will look for masks in those faces only. This helps in achieving an interesting accuracy of 92% on validation Dataset. Due to GPU constraints, We have trained on smaller sized images however We recommend to train on bigger images for more accurate results. We have used MobileNetv2 for fast processing. The model has been implemented on Keras Framework for Python using Tensorflow Backend. During training, we used the annotated objects to get more faces to train. During prediction, the faces will be provided solely using Face Detection MTCNN.

Tools

Python3:

The whole project is written on the beautiful programming language Python. We recommend working on version 3.7 for optimal performance

Tensorflow + Keras:

The face detection part was made easy using the Keras framework. All the standards of Tensorflow 2 have been used in the code.

MTCNN:

We have used the open-source implementation of MTCNN. We thank the developers to provide so easy-to-use library. All the face detection tasks have been accomplished using this.

Tkinter:

Our GUI makes the code usable by anyone and without Tkinter, this implementation couldn’t have been possible. With all the buttons, dialog boxes, Our GUI is a complete package.

PIL and OpenCV:

Both these Image Processing Libraries have been extensively used in all the tasks. These libraries make Image Processing in Python a child’s play. We have tried to exploit many features of these libraries.