Face Detector Streamlit app

CV Project

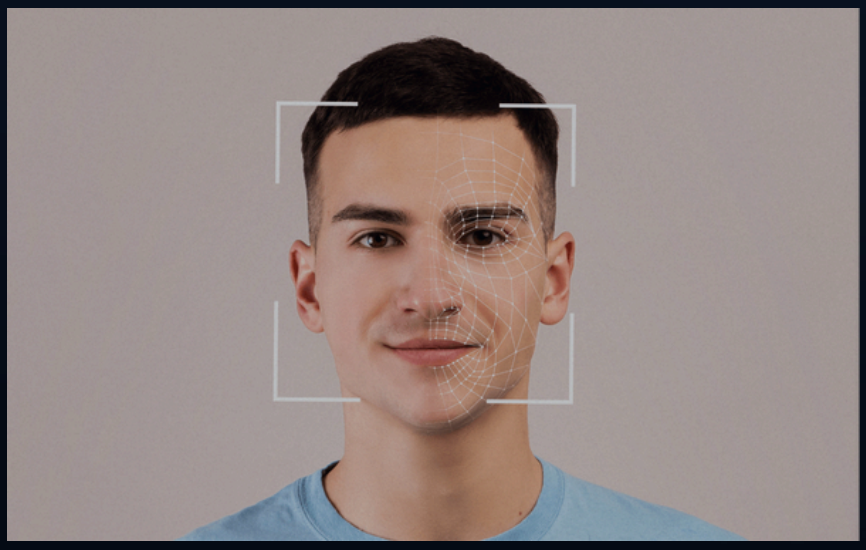
Repo Link:

https://github.com/salmamuhammede/Face-Detection-and-Blurring-Streamlit-app.git

25/06/2024

PROJECT IDEA

The project idea revolves around a Face Detector app using Streamlit and OpenCV. Users can upload an image, and the app detects faces using a pre-trained Haar Cascade classifier. It outlines detected faces with rectangles and applies Gaussian blur to anonymize them. The app provides visual feedback with the original image showing detected faces and another version with blurred faces. This tool can be useful for privacy protection in images or enhancing security features in applications requiring facial recognition and anonymization.



LIBRARIES

Streamlit:

- Application: Streamlit is used in the creation of interactive web applications for machine learning and data science projects.I use it to display photographs, download photos, and upload files.

OpenCV (cv2):

- Usage: OpenCV is a powerful library for computer vision tasks. In this project, it's used for image manipulation, specifically for:
  - Converting between different color spaces (cv2.cvtColor).
  - Loading pre-trained Haar Cascade classifiers (cv2.CascadeClassifier) for face detection.
  - Drawing rectangles around detected faces (cv2.rectangle).
  - Applying Gaussian blur (cv2.GaussianBlur) to anonymize detected faces.

NumPy:

  - Usage: NumPy is a fundamental package for numerical computing in Python.

PIL (Python Imaging Library) / Pillow:

- Usage: PIL (or Pillow) is used for handling images in Python. It's used to load and convert images into a format suitable for processing by OpenCV

Io

METHODOLOGY

Welcome Message and Imports:

- The app starts with a welcome message using streamlit for creating an interactive web application.
- Necessary libraries are imported: cv2 for computer vision tasks, numpy for numerical operations, PIL for image handling, and io for byte conversions.

Model Loading and Variables Initialization:

- The Haar Cascade classifier (haarcascade_frontalface_default.xml) is loaded using OpenCV (cv2.CascadeClassifier).
- Variables (image, grayedImage, blurredImage, detectedImage) are initialized for image processing and storage.

File Uploading and Face Detection:

- Users upload an image file (jpeg, jpg, png) using Streamlit's file_uploader.
- The detectFace function converts the uploaded image to grayscale, detects faces using the loaded Haar Cascade classifier, draws rectangles around detected faces, and applies Gaussian blur to each detected face region.

METHODOLOGY

Image Processing and Display:

- In the process function, the uploaded image is converted to RGB format for compatibility with Streamlit.
- Faces in the original and blurred versions are displayed using st.image in RGB format to ensure correct color representation.

Download Functionality:

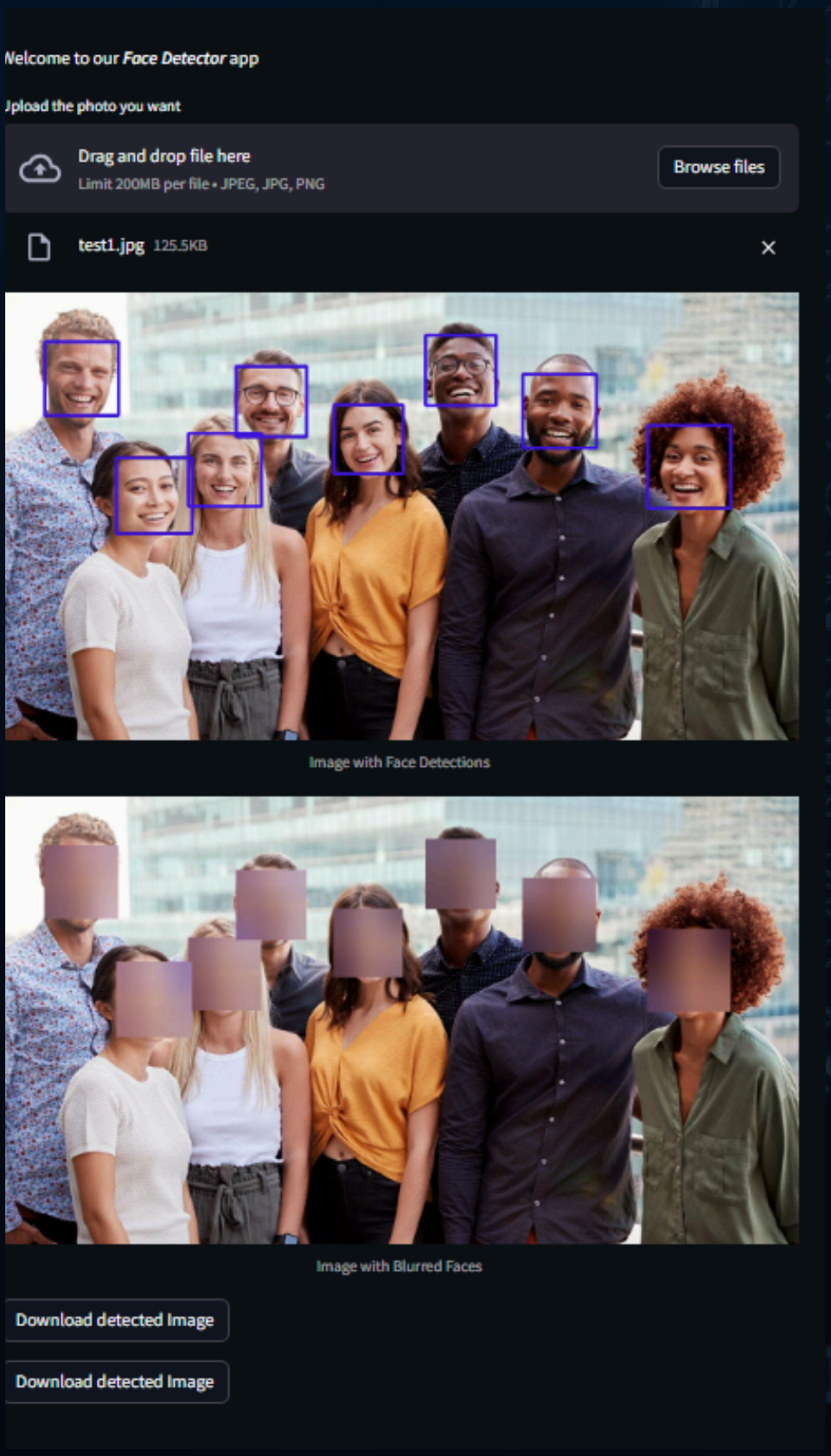
- Two separate download functions (download_Dimage for detected images and download_Bimage for blurred images) are defined.
- These functions convert the processed images back to PIL format, save them as JPEG files in memory (io.BytesIO), convert them to bytes (image_io.getvalue()), and create download buttons (st.download_button) with appropriate labels and MIME types (image/jpeg).

User Interaction and Output:

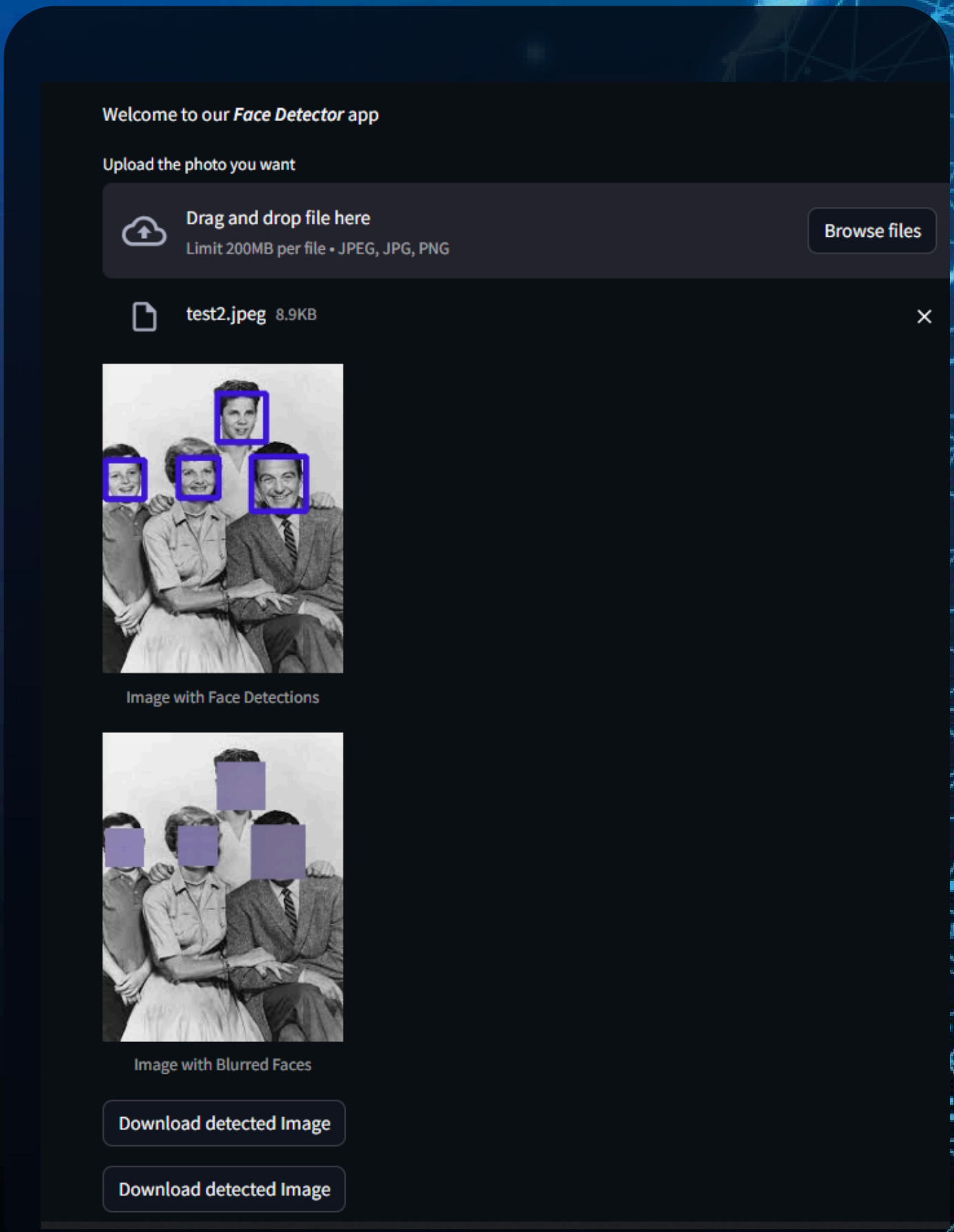
- After processing and displaying images with face detections and blurring effects, the app checks if valid images are available (if detectedImage is not None and if blurredImage is not None).
- It then offers download buttons (st.download_button) for both the detected and blurred images, ensuring users can save these processed images locally.

RESULT SNAPSHOTS

RESULT SNAPSHOTS



RESULT SNAPSHOTS



CHALLENGES

Experience with Computer Vision Models:

- Challenge: Encountered issues with color conversion during image processing.
- Solution: Studied the Haar Cascade model thoroughly through YouTube tutorials to understand its parameters and functionality, ensuring proper handling of image types and color conversions.

Learning and Application of Haar Cascade:

- Process:
  - Focused on understanding the Haar Cascade model, its parameters, and how to effectively use it for face detection.
  - Learned about the importance of accurate color space conversions between RGB and BGR formats to avoid color discrepancies.
- Outcome: Gained proficiency in using the Haar Cascade model for detecting faces without color conversion issues.

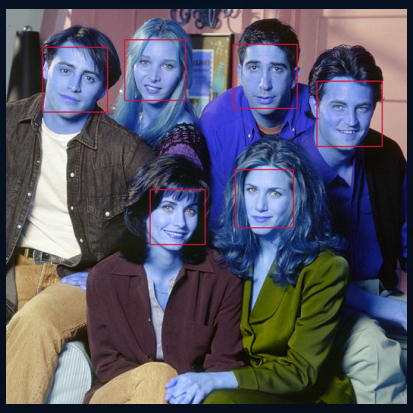
Proficiency with Streamlit:

- Experience: Comfortable using Streamlit from previous work on an object detection app.
- Streamlit Specifics: No significant challenges faced while integrating Streamlit for the face detection app, owing to prior experience.

CHALLENGES

Issue with Downloading Processed Images:

- Challenge: Faces appeared blue in the downloaded images due to improper color space handling during the conversion process.



- Solution:
  - Researched solutions and discovered that the issue was related to incorrect handling of color spaces when converting images for download.
  - Implemented the use of the io library to handle image conversions correctly:
    - Used PIL to convert images to the correct format.
    - Used io.BytesIO() to save the image as bytes and then provide the correct format and color space for downloading.
- Outcome: Successfully resolved the issue, ensuring that downloaded images maintained the correct colors.