My Data Talk

Create A Photo Editing App Using Streamlit — Surprisingly Easy and Fun

Let’s Have Some Fun with Streamlit and Computer Vision: A Hands-on Guide for Beginners

Have you ever used one of those photo editing apps that allow you to upload your photo online and instantly convert it to a black-and-white image, a cool pencil sketch, or a picture with a nice blurring effect? As a curious data scientist, I am always interested in finding out how those apps were created behind the scene. However, for a long time, I haven't really devoted any time to learning and trying it myself.

Recently, I came across a few articles online that talk about how to use OpenCV, a real-time optimized Computer Vision library in Python, to turn a photo into a sketch, or a gray-scale image, etc. I was really amazed by how easy and straightforward it is to use OpenCV to get started with some of the simple tasks in the world of computer vision.

In the meantime, I have been exploring and learning Streamlit for a while. It is a free, open-source, all-python framework that enables data scientists to quickly build data and machine learning web apps with no front-end web development experience required. So suddenly something clicked — this could be a perfect hands-on Streamlit project for me to build my own photo editing app with OpenCV!

In less than two hours and with only about sixty lines of code I was able to build a photo editing app using Streamlit and it looks pretty good to me! I'd like to share my project work with you so that you can also have some fun learning to build your own photo editing app! The app we are going to build looks like this (or watch this short YouTube Video Demo):

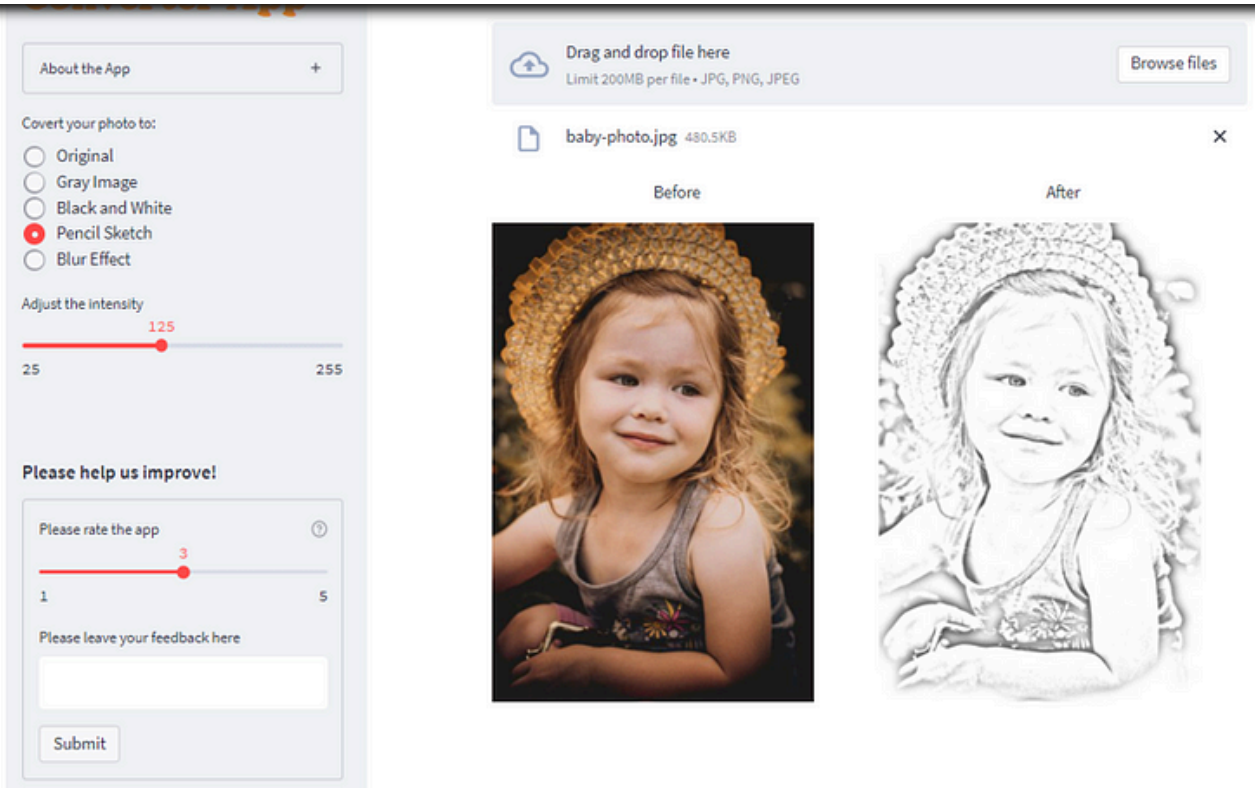


Image by Author (Baby Photo by Pixabay)

Pre-requisite

If you haven't already done so, you need to install Streamlit and OpenCV on your computer for this project.

#1: Installation of Streamlit:

You can refer to the following article and follow the instructions to install Streamlit and learn the basics.

Streamlit Hands-On: From Zero to Your First Awesome Web App

Installation of OpenCV:

You can use the following command to install OpenCV or refer to its documentation page for more details: https://pypi.org/project/opencv-python/

```
pip install opencv-python
```

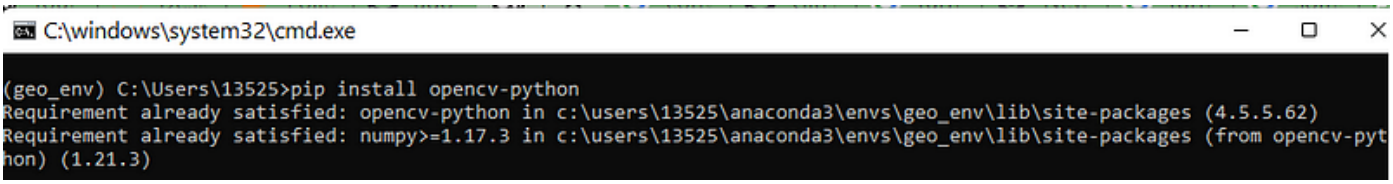


Image by Author

Launch Streamlit

Let's open the VS code editor (or any text editor of your choice), create a new empty python file, and save it as photo_converter_app.py in your project folder. Then we can fire up Streamlit from the Anaconda terminal window. A blank Streamlit app should appear in your local web browser.

```
streamlit run photo_converter_app.py
```

```
Local URL: http://localhost:8501
Network URL: http://192.168.1.159:8501
```

Image by Author

Import Libraries

Let's begin our app-building journey by first importing all the necessary libraries:

```
#Import libraries
import streamlit as st
import numpy as np
import cv2
from  PIL import Image, ImageEnhance
```

Import libraries.py hosted with ❤ by GitHub view raw

Add Header, Brand Logo, and Sidebar to the App

In the main interface of the app, we want to add a header using st.markdown() and also an optional brand logo. The reason we choose st.markdown() instead of st.title() is because we can use CSS to style it and make it more appealing.

```
#Create two columns with different width
col1, col2 = st.columns( [0.8, 0.2])
with col1:                # To display the header text using css style
    st.markdown(""" <style> .font {
    font-size:35px ; font-family: 'Cooper Black'; color: #FF9633;}
    </style> """, unsafe_allow_html=True)
    st.markdown('<p class="font">Upload your photo here...</p>', unsafe_allow_html=True)

with col2:                # To display brand logo
    st.image(image,  width=150)
```

Add header and brand logo to the app interface.py hosted with ❤ by GitHub view raw

Let's also add a header and expander in the sidebar to provide more information about the app by using the code below:

```
#Add a header and expander in side bar
st.sidebar.markdown('<p class="font">My First Photo Converter App</p>', unsafe_allow_html=True)
with st.sidebar.expander("About the App"):
    st.write("""
       Use this simple app to convert your favorite photo to a pencil sketch, a grayscale image
    """)
```

Add a header and expander in side bar.py hosted with ❤ by GitHub view raw

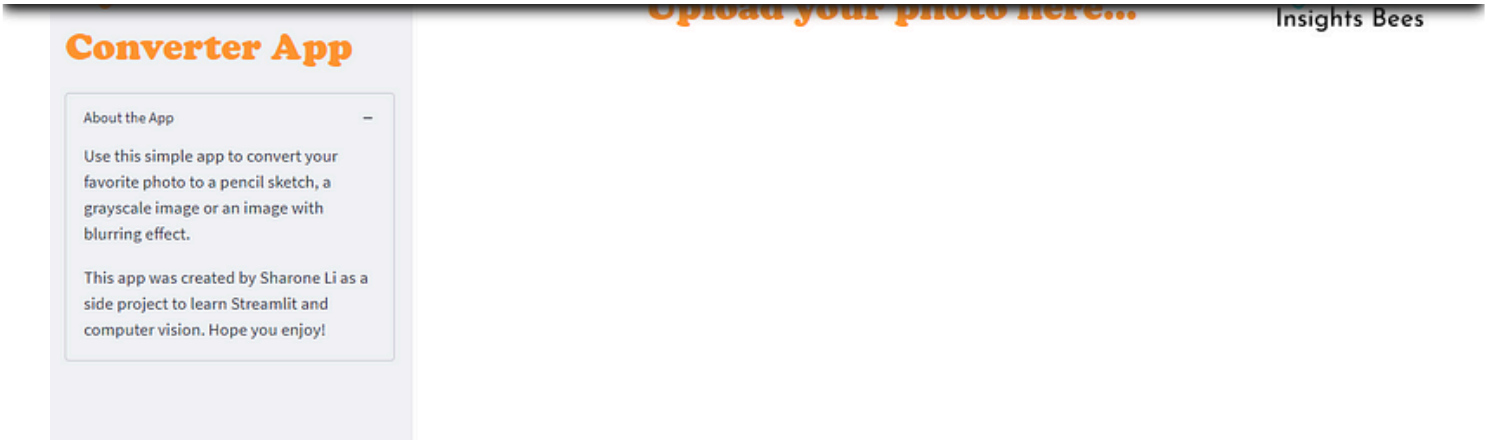


Image by Author

Add File Uploader to Allow Users to Upload Photos

In the main interface of the app, we want to add a file uploader so that users can upload their photos by either drag-and-drop or browsing files. We can use the st.file_uploader() widget to do that and specify the image types that are accepted in the app(e.g., JPG, PNG, JPEG, etc.)

```
#Add file uploader to allow users to upload photos
uploaded_file = st.file_uploader("", type=['jpg','png','jpeg'])
```

Add st.file_uploader() widget.py hosted with ❤ by GitHub view raw

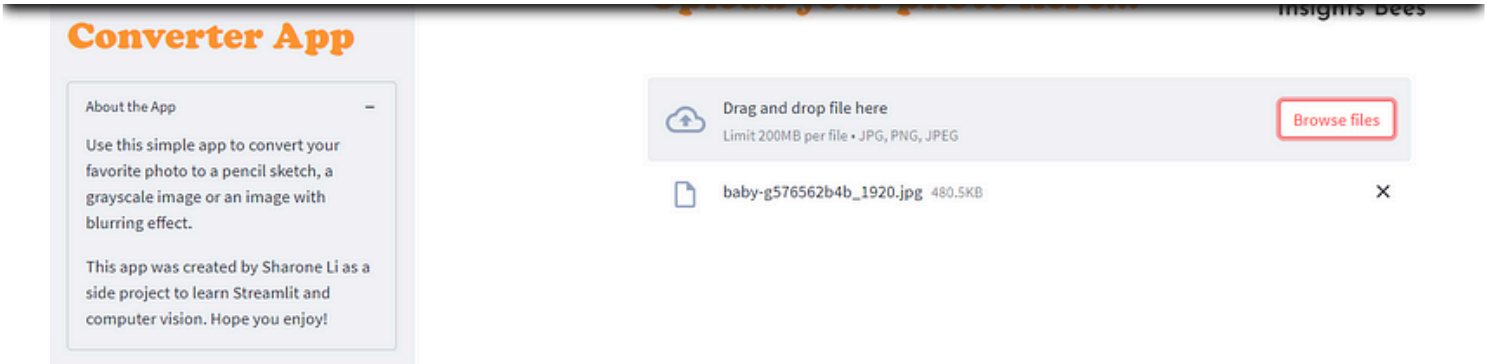


Image by Author

Add Space Holder to Show Before vs. After Images

After users upload a photo, we want to show the original image (before) and the converted image (after), side by side, in the app. Therefore, let's create two columns with the same width under the file uploader, one for 'before' and the other for 'after'. Make sure you put everything within the '*if uploaded_file is not None:*' statement. Using this statement ensures that when an image hasn't been uploaded the app doesn't throw any error message for not finding any images to convert.

```
#Add 'before' and 'after' columns
if uploaded_file is not None:
    image = Image.open(uploaded_file)

    col1, col2 = st.columns( [0.5, 0.5])
    with col1:
        st.markdown('<p style="text-align: center;">Before</p>',unsafe_allow_html=True)
        st.image(image,width=300)

    with col2:
        st.markdown('<p style="text-align: center;">After</p>',unsafe_allow_html=True)
```

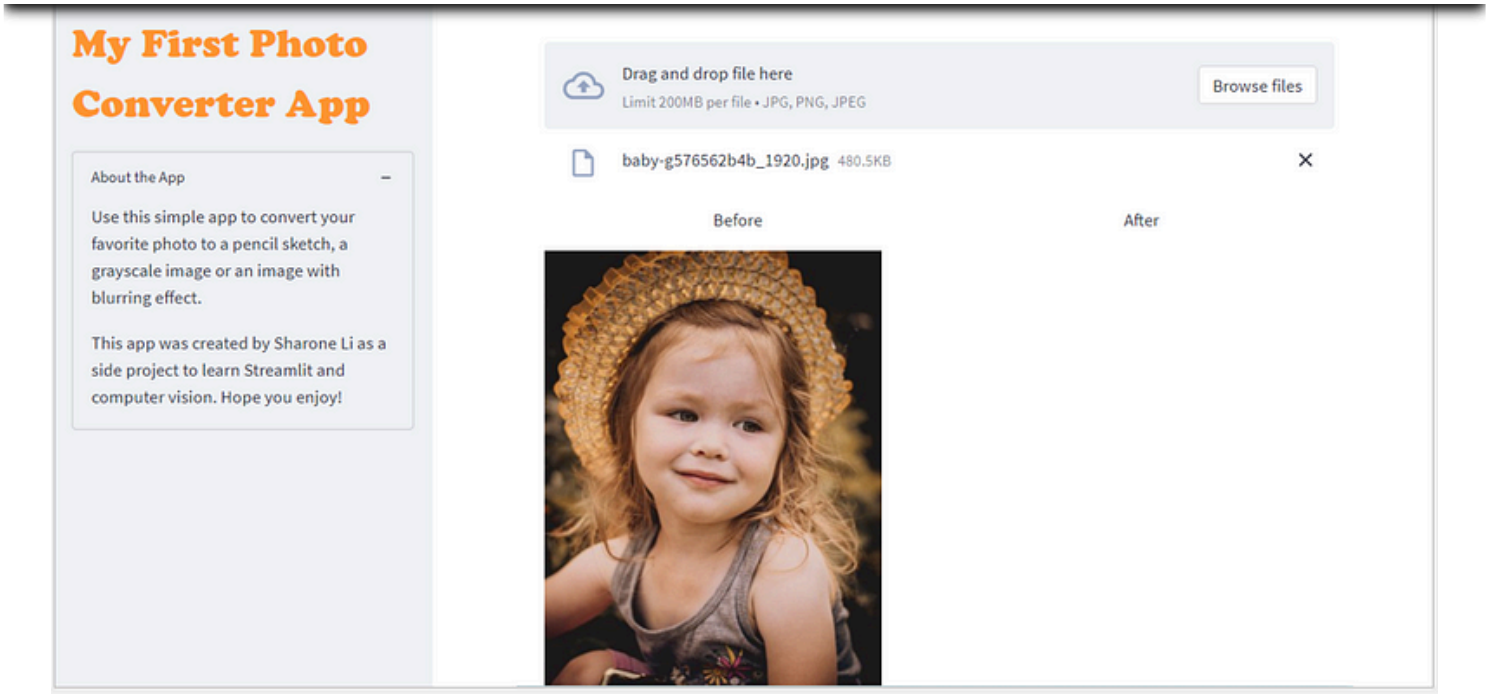


Image by Author

Use OpenCV to Convert Photos

Now comes the fun part! In the second column col2 (the place holder for the 'after' image), we want to display the converted image based on the user's input. For example, if a user wants to convert the image to black-and-white, we will convert the original image to black-and-white using OpenCV. If the user wants to convert it to a pencil sketch, we will use OpenCV to convert it to a pencil sketch. How can we achieve that?

#1: Create a Filter that Takes User Input

We'll first need to create a filter or single-select box to allow users to specify what they want to do. To keep the main interface of the app clean, we can add

Notice that we placed the filter variable inside the '*if uploaded_file is not None:*' code block. By doing so the filter will only appear after users upload an image. You can also choose to place the filter variable outside of the code block (i.e., above line 2). That way the filter will always show in the sidebar. It's really just a personal preference for the app design.

```
1   uploaded_file = st.file_uploader("", type=['jpg','png','jpeg'])
2   if uploaded_file is not None:
3       image = Image.open(uploaded_file)
4
5       col1, col2 = st.columns( [0.5, 0.5])
6       with col1:
7           st.markdown('<p style="text-align: center;">Before</p>',unsafe_allow_html=True)
8           st.image(image,width=300)
9
10      with col2:
11          st.markdown('<p style="text-align: center;">After</p>',unsafe_allow_html=True)
12          filter = st.sidebar.radio('Covert your photo to:', ['Original','Black and White', 'Penci
```

Add the filter in the sidebar.py hosted with ❤ by GitHub view raw

#2: Use Conditional Statements to Take User Input

You will see the radio button filter appear immediately in the sidebar; however, when you click an option, it won't trigger anything in the app to show the converted image. This is because we haven't told Streamlit where to pass the values of the filters and what action the user's input triggers. We'll need to add

we use different OpenCV functions (e.g., cv2.cvtColor, cv2.cv2.GaussianBlur, cv2.divide, etc.) to convert the image to the desired format and display the converted image using st.image().

```
#Add conditional statements to take the user input values
    with col2:
        st.markdown('<p style="text-align: center;">After</p>',unsafe_allow_html=True)
        filter = st.sidebar.radio('Covert your photo to:', ['Original','Gray Image','Black and W
        if filter == 'Gray Image':
                converted_img = np.array(image.convert('RGB'))
                gray_scale = cv2.cvtColor(converted_img, cv2.COLOR_RGB2GRAY)
                st.image(gray_scale, width=300)
        elif filter == 'Black and White':
                converted_img = np.array(image.convert('RGB'))
                gray_scale = cv2.cvtColor(converted_img, cv2.COLOR_RGB2GRAY)
                slider = st.sidebar.slider('Adjust the intensity', 1, 255, 127, step=1)
                (thresh, blackAndWhiteImage) = cv2.threshold(gray_scale, slider, 255, cv2.THRESH
                st.image(blackAndWhiteImage, width=300)
        elif filter == 'Pencil Sketch':
                converted_img = np.array(image.convert('RGB'))
                gray_scale = cv2.cvtColor(converted_img, cv2.COLOR_RGB2GRAY)
                inv_gray = 255 - gray_scale
                slider = st.sidebar.slider('Adjust the intensity', 25, 255, 125, step=2)
                blur_image = cv2.GaussianBlur(inv_gray, (slider,slider), 0, 0)
                sketch = cv2.divide(gray_scale, 255 - blur_image, scale=256)
                st.image(sketch, width=300)
        elif filter == 'Blur Effect':
                converted_img = np.array(image.convert('RGB'))
                slider = st.sidebar.slider('Adjust the intensity', 5, 81, 33, step=2)
                converted_img = cv2.cvtColor(converted_img, cv2.COLOR_RGB2BGR)
                blur_image = cv2.GaussianBlur(converted_img, (slider,slider), 0, 0)
                st.image(blur_image, channels='BGR', width=300)
        else:
                st.image(image, width=300)
```

Let's examine the code above in greater detail and understand how OpenCV works in converting images to different formats:

Line 5–8:

Converting an image to a gray image is really straightforward using OpenCV. We can simply use the cvtColor() function and specify the color space conversion code to be cv2.COLOR_BGR2GRAY.

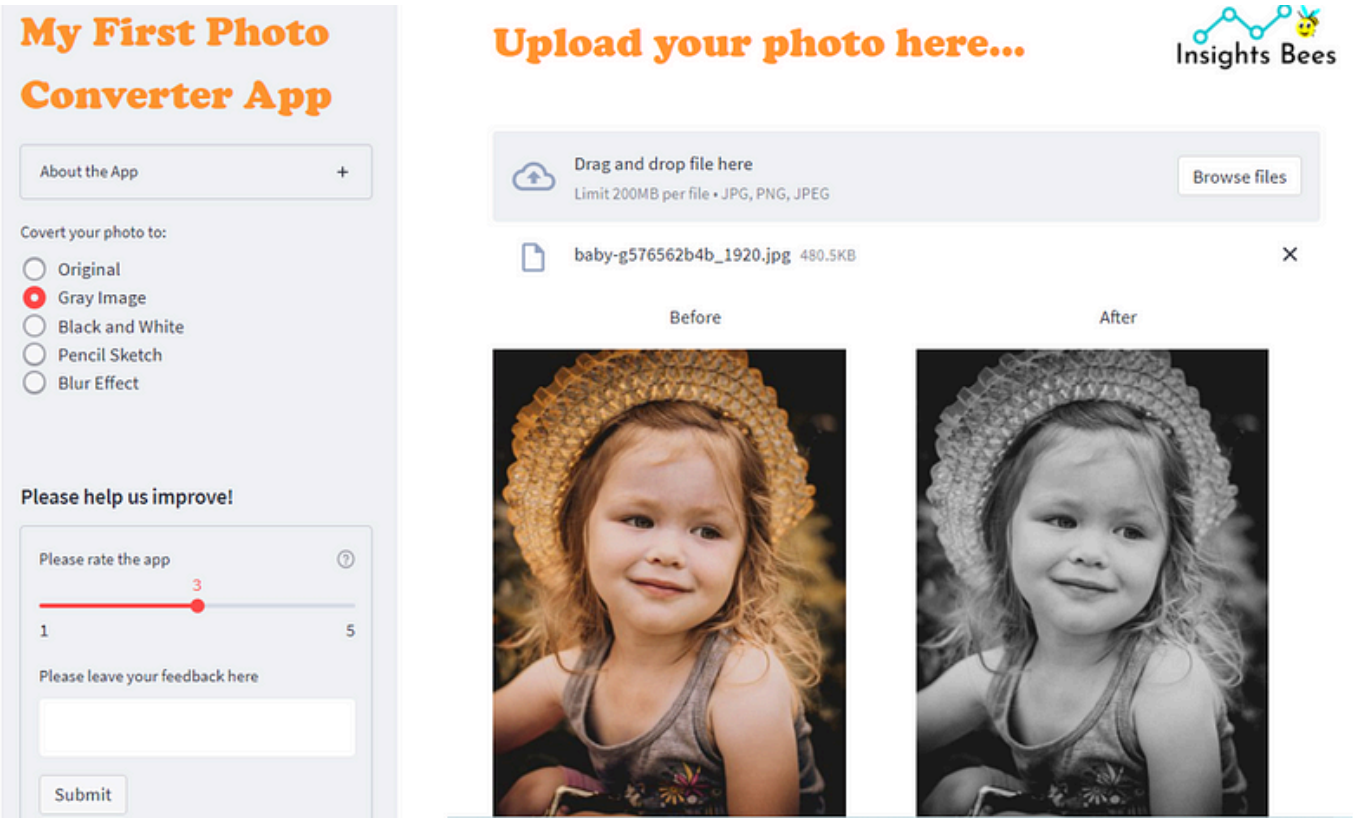


Image by Author

Line 9–14:

each pixel of the grayscale image, we assign the value of 0 (black) to the pixel if its value is below the threshold or the value 255 (white) if its value is above the threshold.

Notice that in line 12, we created a slider widget that allows users to pick a threshold, and line 13 takes the threshold value and converts the gray image to a black-and-white image.

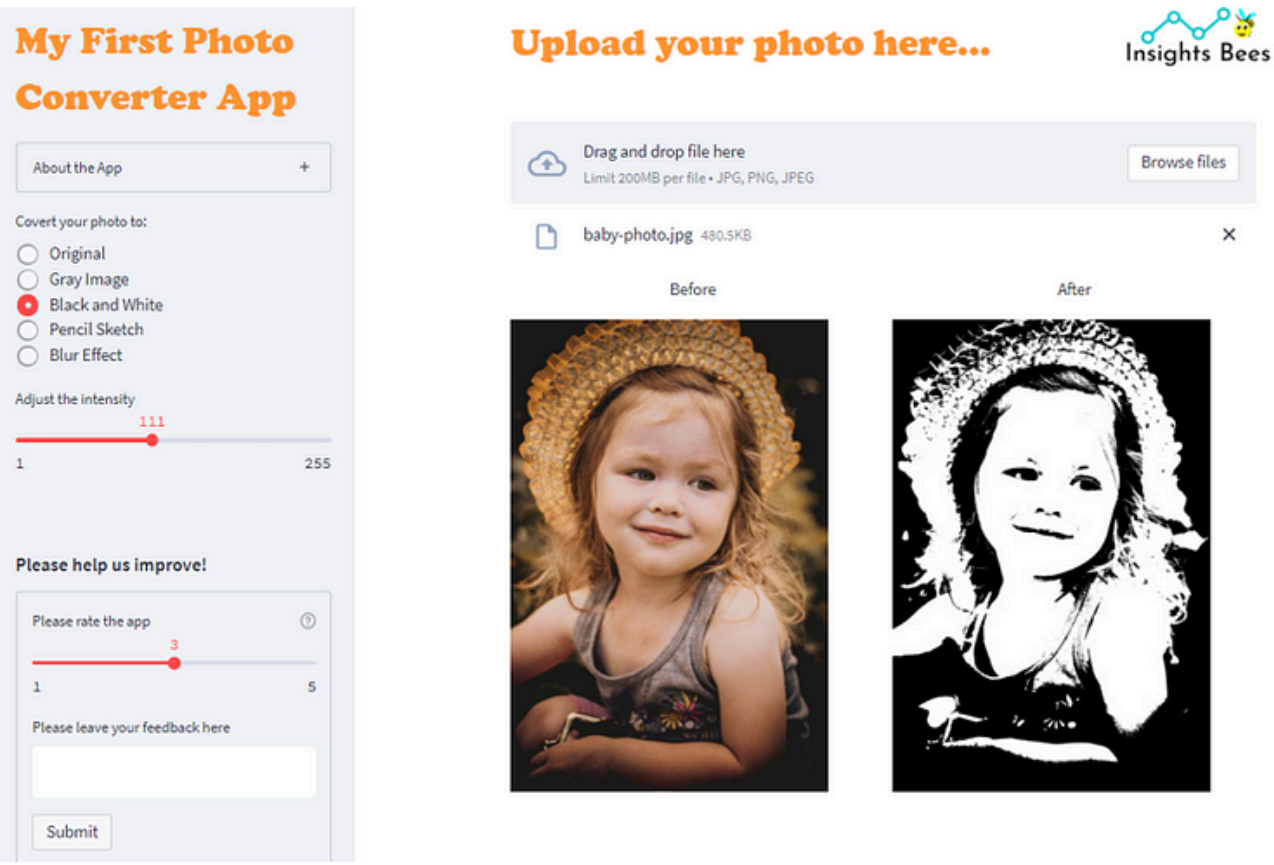


Image by Author

Line 15–22:

pick the Gaussian Kernel Size to adjust the intensity of the blur. We choose (125,125) as the default values of the Kernel Size parameters([height width]. The height and width should be odd numbers and can have different values.

Lastly, we use the cv2.divide function to divide the pixels in the gray image with those in the 255-blur_image. This returns an image that looks like a pencil drawing.

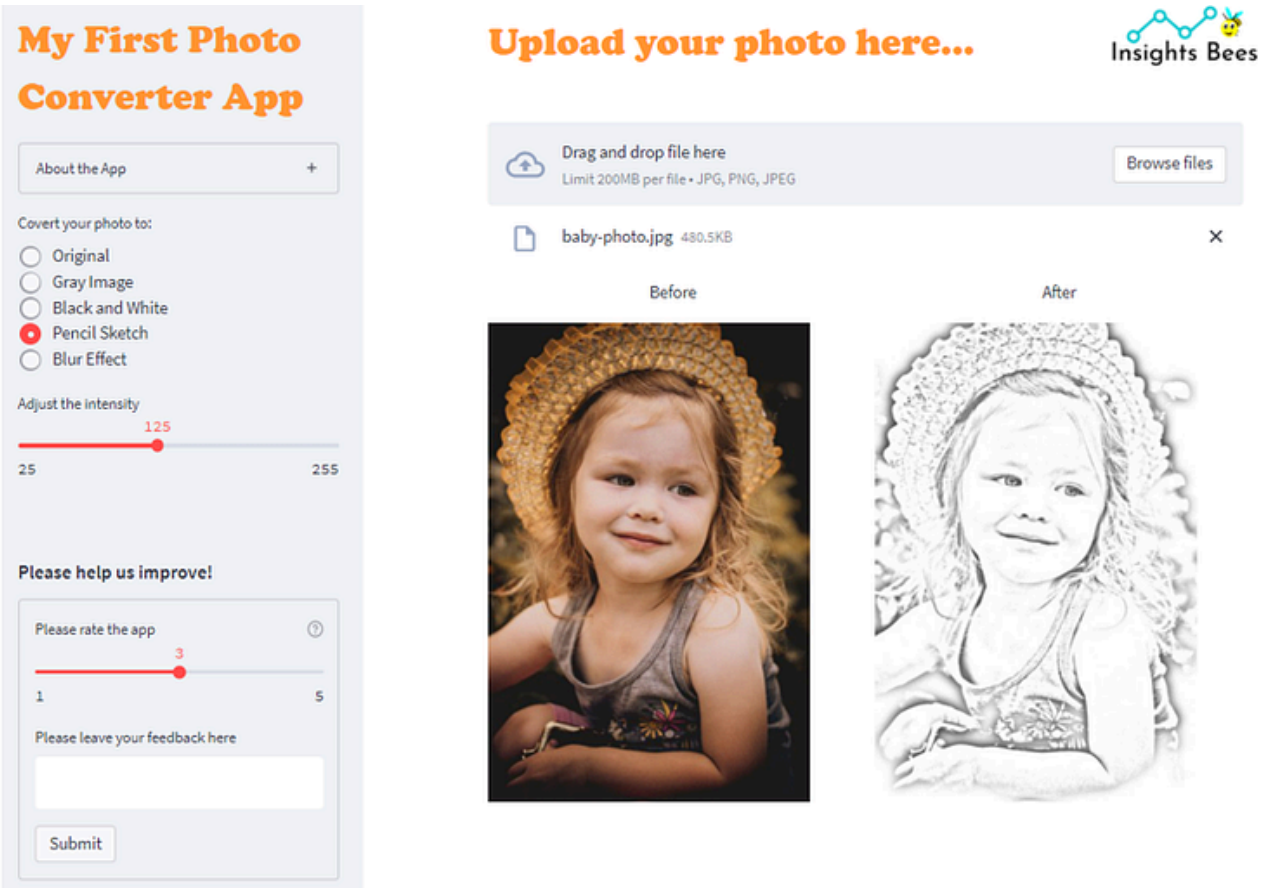


Image by Author

Line 23–28:

(cv2.GaussianBlur). Notice that in line 25 we also added a conditional Streamlit widget st.siderbar.slider() which appears only when the user selects the 'Blur Effect' option and allows the user to adjust the intensity of blur accordingly.

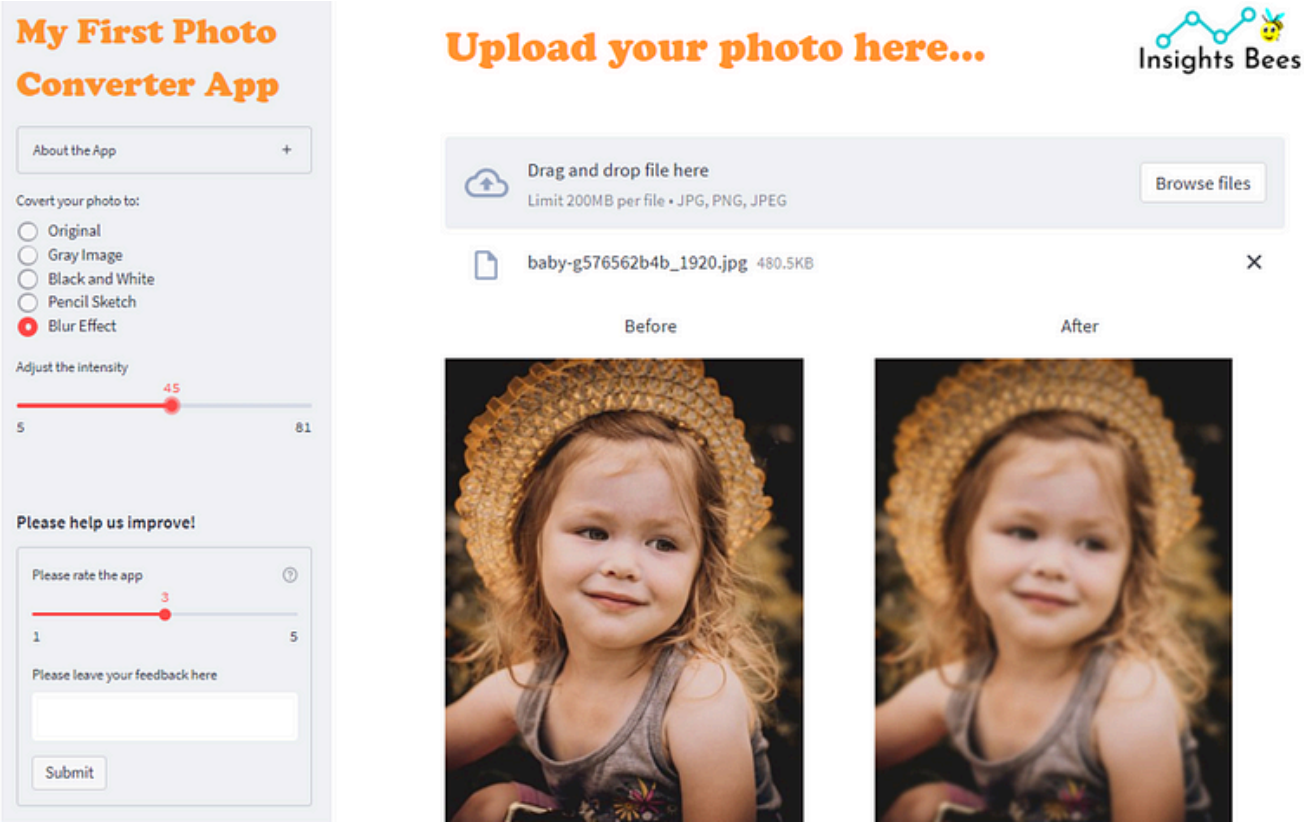


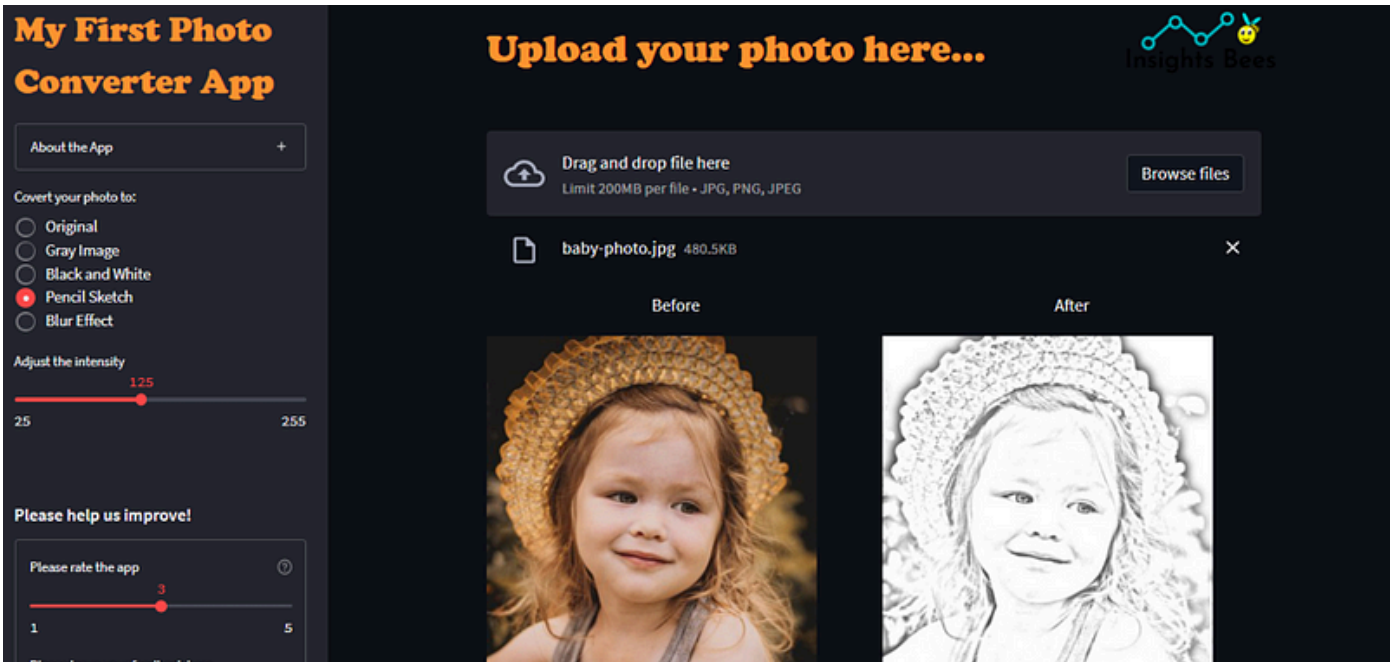
Image by Author

Add a User Feedback Section in the Sidebar

Finally, we'll add a user feedback section in the sidebar to collect review ratings and comments. We use the st.text_input() widget to allow users to submit comments and use the st.slider() widget to let users select a rating on a scale of

```python
#Add a feedback section in the sidebar
st.sidebar.title(' ') #Used to create some space between the filter widget and the comments sect
st.sidebar.markdown(' ') #Used to create some space between the filter widget and the comments s
st.sidebar.subheader('Please help us improve!')
with st.sidebar.form(key='columns_in_form',clear_on_submit=True): #set clear_on_submit=True so t
    rating=st.slider("Please rate the app", min_value=1, max_value=5, value=3,help='Drag the sli
    text=st.text_input(label='Please leave your feedback here')
    submitted = st.form_submit_button('Submit')
    if submitted:
      st.write('Thanks for your feedback!')
      st.markdown('Your Rating:')
      st.markdown(rating)
      st.markdown('Your Feedback:')
      st.markdown(text)
```

Add a feedback section in the sidebar.py hosted with ❤ by GitHub view raw



There you go with your first photo editing app using Streamlit! You can also share your app with others via Streamlit Cloud, a very cool capability launched by Streamlit recently. By being hands-on and working on a project like this, I was able to level up my Streamlit skills and get my first step into the world of computer vision. I truly enjoyed this project and hope you had fun too! Happy Learning!

Here is a short YouTube video to demo the app:

- Meet Streamlit Sharing | Build a Simple Photo Editor by Rafael Messias Grecco
- Turn Your Photos into Artistic Sketches with Code by Behic Guven
- Image Source: the baby photo used in the app can be downloaded from Pixabay. It is free for commercial use (No attribution required).

You can unlock full access to my writing and the rest of Medium by signing up for Medium membership ($5 per month) through this referral link. By signing up through this link, I will receive a portion of your membership fee at no additional cost to you. Thank you!

Streamlit | Data Science | Python | Data Visualization | AI

Lakmina Pramodya Gamage

Create your own PDF chatbot with OpenAI, Langchain and Streamlit

Chatbots has become a trend of nowadays due to the rise of LLM (large language models) products like ChatGPT, Bing AI, Bard AI etc. In this...

6 min read

Sujatha Mudadla

Differences between Streamlit and Gradio.

Streamlit:

2 min read

Bill Parker

Build a Chatbot Application with Ollama and Open Source Models

Creating a chat application that is both easy to build and versatile enough to integrate with open source large language models or...

5 min read

Hassan Sheikh

Simple Survey Using Streamlit + SQLite

Streamlit is a gateway to designing interactive web apps with fantastic UIs. In this article, I used its simplicity to craft a sample...

7 min read