Naan Mudhalvan Project Documentation

-Joel Ananth A

2021503022

COLORIZATION OF BLACK AND WHITE PICTURES

INTRODUCTION

This Python script allows you to colorize black and white images using deep learning techniques. It utilizes the OpenCV library for computer vision and the Tkinter library for the graphical user interface (GUI).

PROBLEM STATEMENT

Black and white images often lack the visual richness of their colored counterparts, limiting their aesthetic appeal and historical context. Manually colorizing these images can be time-consuming and requires expertise. Thus, there is a need for an automated solution that can efficiently colorize black and white images while preserving their authenticity.

PROJECT OVERVIEW

This project aims to develop a Python script utilizing deep learning techniques to automatically colorize black and white images. The script utilizes the OpenCV library for computer vision and the Tkinter library for the graphical user interface (GUI). By leveraging pre-trained models and image processing algorithms, the script transforms grayscale images into vibrant colorized versions.

END USERS

1. Historians and archivists: To enhance the visual representation of historical photographs and documents.
2. Photographers and artists: To experiment with different color schemes and enhance the artistic expression of their work.
3. General users: To effortlessly add color to black and white images for personal or professional use.

SOLUTION AND ITS VALUE PROPOSITION

The proposed solution offers the following key benefits:

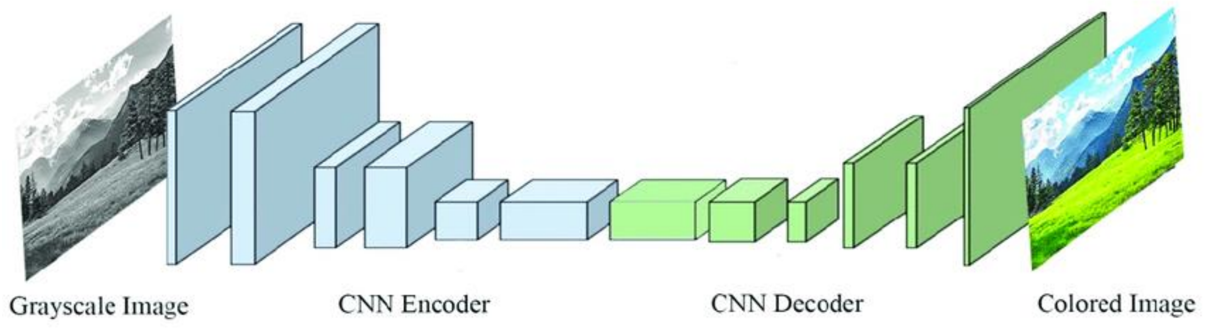
- Automation: Eliminates the need for manual colorization, saving time and effort.
- Accessibility: Provides a user-friendly interface for individuals with limited technical expertise.
- Preservation: Helps preserve and revive historical black and white images by adding color.

- Creativity: Allows users to experiment with different colorization techniques and enhance their creative projects.

NOVELTY

- Integration of deep learning techniques: Utilizes pre-trained models and advanced algorithms for accurate colorization.
- Graphical user interface (GUI): Enhances user experience by providing a visually appealing and intuitive interface for image colorization.
- Seamless workflow: Streamlines the process of uploading, colorizing, and viewing images within a single application.

ARCHITECTURE DIAGRAM



REQUIREMNETS

Make sure you have the following dependencies installed:

Python 3.x

Tkinter

NumPy

OpenCV (cv2)

Pillow (PIL)

The Python Imaging Library (PIL)

matplotlib

You can install these dependencies using pip:

```
pip install numpy opencv-python pillow matplotlib
```

USAGE

Clone or download the repository to your local machine.

Ensure that your black and white images are in a compatible format (e.g., PNG, JPEG).

Run the Python script 'main.py'.

Use the GUI to upload a black and white image by clicking on "Upload Image".

Once the image is uploaded, click on "Color Image" to colorize it.

The colorized image will be displayed in the GUI.

FILES

- colorize_images.py: Python script for colorizing black and white images.
- models/pts_in_hull.npy: Numpy file containing pre-calculated points for colorization.
- models/colorization_deploy_v2.prototxt: Model configuration file for colorization.
- models/colorization_release_v2.caffemodel: Pre-trained model weights for colorization.
- logo2.png: Logo image used in the GUI.

NOTES

- Make sure to place the pts_in_hull.npy, colorization_deploy_v2.prototxt, and colorization_release_v2.caffemodel files in the models directory.
- The logo2.png file is used as the logo in the GUI. You can replace it with your own logo if desired.
- Ensure that your Python environment has the required libraries installed before running the script.
- The script resizes images to a fixed size (480x360) for processing. You can modify this size according to your requirements.
- The colorized image will be saved as result.png in the current directory.

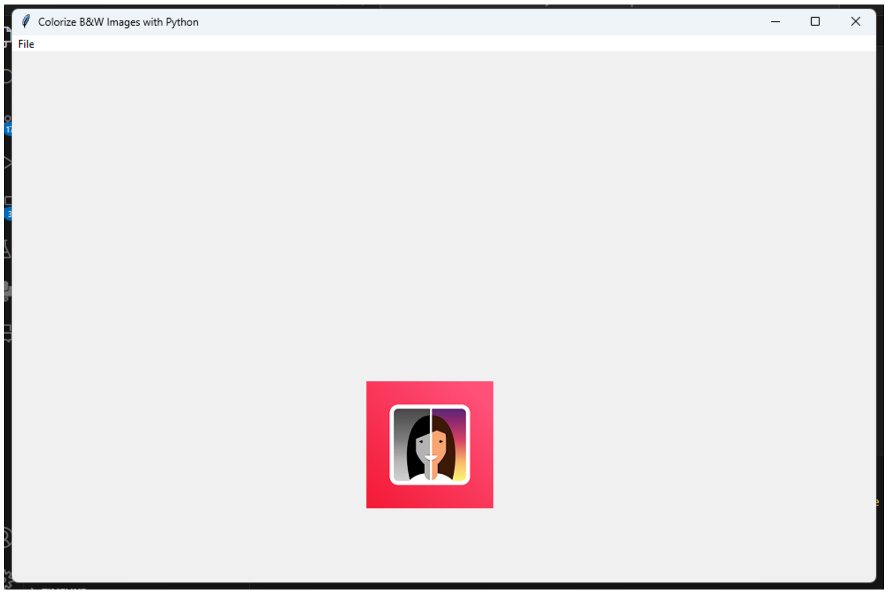
PROGRAM/CODE

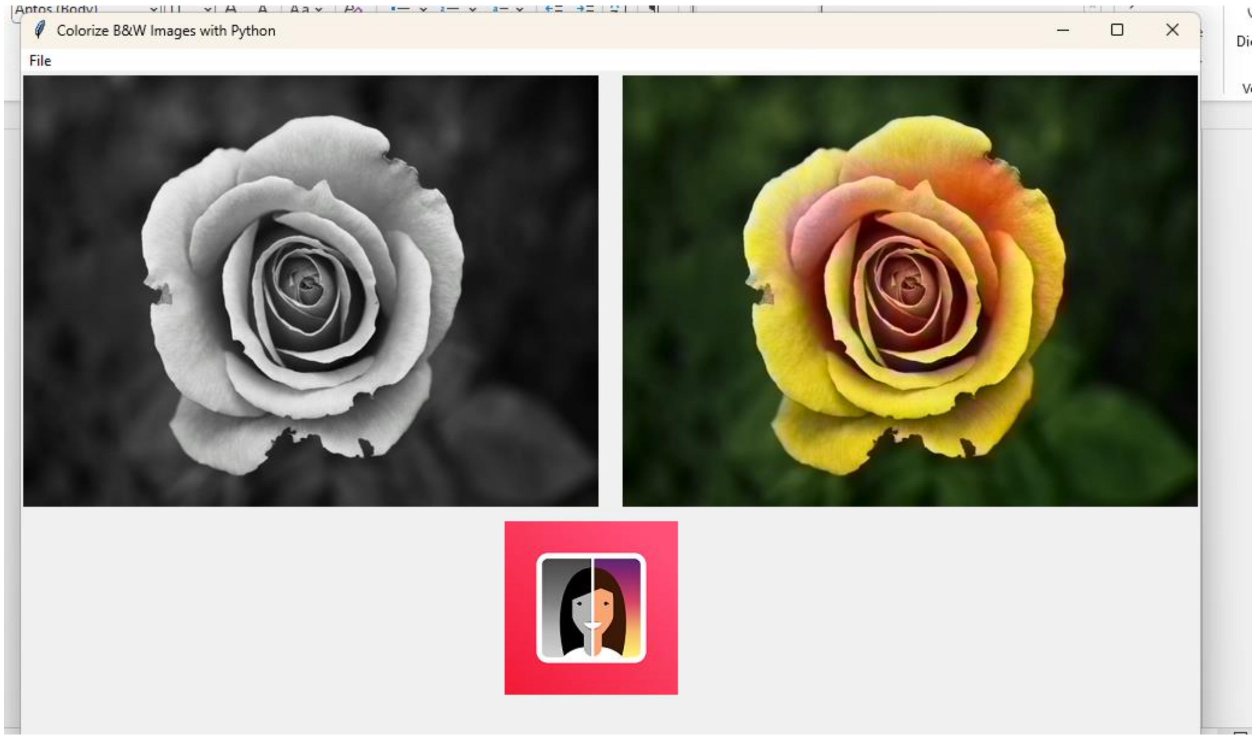
```
import tkinter as tk
from tkinter import *
from tkinter import filedialog
from PIL import Image,ImageTk
```

```
import os
import numpy as np
import cv2 as cv
import os.path

numpy_file = np.load('./models/pts_in_hull.npy')
Caffe_net = cv.dnn.readNetFromCaffe("./models/colorization_deploy_v2.prototxt",
                                    "./models/colorization_release_v2.caffemodel")
numpy_file = numpy_file.transpose().reshape(2, 313, 1, 1)


class Window(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
        self.pos = []
        self.master.title("Colorize B&W Images with Python")
        self.pack(fill=BOTH, expand=1)
        menu = Menu(self.master)
        self.master.config(menu=menu)
        file = Menu(menu)
        file.add_command(label="Upload Image", command=self.uploadImage)
        file.add_command(label="Color Image", command=self.color)
        menu.add_cascade(label="File", menu=file)
        self.canvas = tk.Canvas(self)
        self.canvas.pack(fill=tk.BOTH, expand=True)
        self.image = None
        self.image2 = None
        label1 = Label(self, image=img)
        label1.image = img
        label1.place(x=400, y=370)

    def uploadImage(self):
        filename = filedialog.askopenfilename(initialdir=os.getcwd())
        if not filename:
            return
        load = Image.open(filename)
        load = load.resize((480, 360), Image.LANCZOS)
        if self.image is None:
            w, h = load.size
            width, height = root.winfo_width(), root.winfo_height()
            self.render = ImageTk.PhotoImage(load)
            self.image = self.canvas.create_image((w / 2, h / 2), image=self.render)
        else:
            self.canvas.delete(self.image3)
            w, h = load.size
            width, height = root.winfo_screenmmwidth(), root.winfo_screenheight()

            self.render2 = ImageTk.PhotoImage(load)
            self.image2 = self.canvas.create_image((w / 2, h / 2), image=self.render2)
        frame = cv.imread(filename)

        Caffe_net.getLayer(Caffe_net.getLayerId('class8_ab')).blobs = [numpy_file.astype(np.float32)]
        Caffe_net.getLayer(Caffe_net.getLayerId('conv8_313_rh')).blobs = [np.full([1, 313], 2.606,
np.float32)]
        input_width = 224
        input_height = 224
        rgb_img = (frame[:, :, [2, 1, 0]] * 1.0 / 255).astype(np.float32)
        lab_img = cv.cvtColor(rgb_img, cv.COLOR_RGB2Lab)
        l_channel = lab_img[:, :, 0]
        l_channel_resize = cv.resize(l_channel, (input_width, input_height))
        l_channel_resize -= 50
        Caffe_net.setInput(cv.dnn.blobFromImage(l_channel_resize))
        ab_channel = Caffe_net.forward()[0, :, :, :].transpose((1, 2, 0))
        (original_height, original_width) = rgb_img.shape[:2]
        ab_channel_us = cv.resize(ab_channel, (original_width, original_height))
        lab_output = np.concatenate((l_channel[:, :, np.newaxis], ab_channel_us), axis=2)
        bgr_output = np.clip(cv.cvtColor(lab_output, cv.COLOR_Lab2BGR), 0, 1)

        cv.imwrite("./result.png", (bgr_output * 255).astype(np.uint8))

    def color(self):
        load = Image.open("./result.png")
        load = load.resize((480, 360), Image.LANCZOS)
        if self.image is None:
            w, h = load.size
            self.render = ImageTk.PhotoImage(load)
            self.image = self.canvas.create_image((w / 2, h / 2), image=self.render)
            root.geometry("%dx%d" % (w, h))
        else:
            w, h = load.size
            width, height = root.winfo_screenmmwidth(), root.winfo_screenheight()
            self.render3 = ImageTk.PhotoImage(load)
            self.image3 = self.canvas.create_image((w / 2, h / 2), image=self.render3)
            self.canvas.move(self.image3, 500, 0)
```

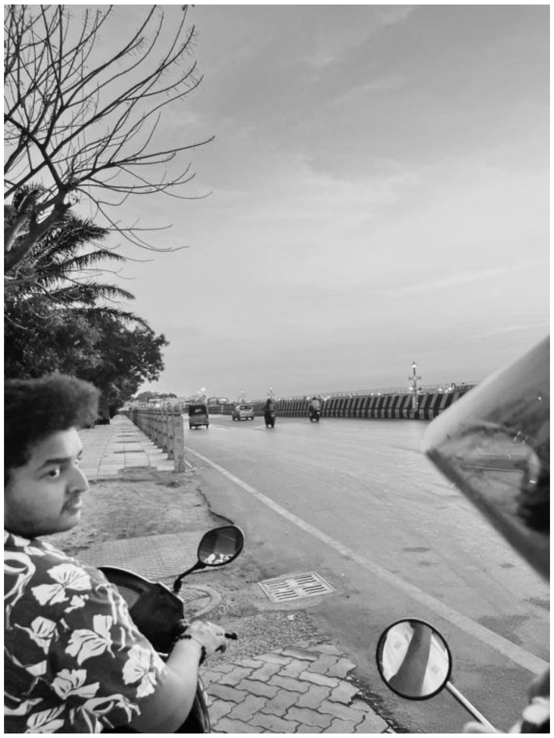
```
root = tk.Tk()
root.geometry("%dx%d" % (980, 600))
root.title("Project-Colorization")
img = ImageTk.PhotoImage(Image.open("logo2.png"))
app = Window(root)
app.pack(fill=tk.BOTH, expand=1)
root.mainloop()
```

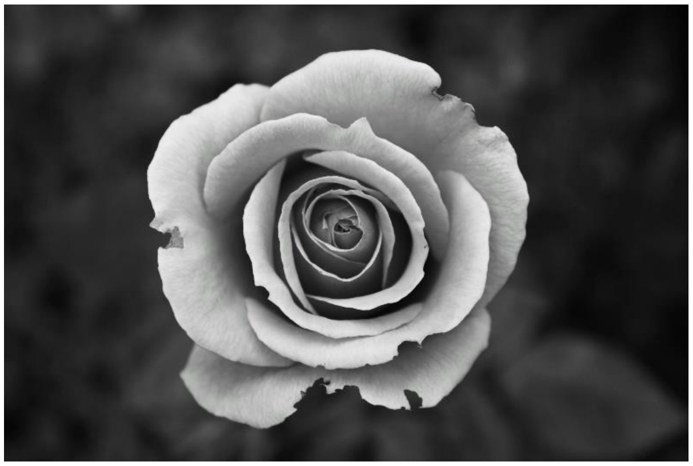
Output Panel:





Input Image:





Output Image: