1.INTRODUCTION

Indeed, the act of coloring a grayscale image is a task that comes naturally to the human mind. From an early age, we learn to fill in missing colors in coloring books, recognizing that grass is green, the sky is blue with white clouds, or that an apple can be red or green. These basic color associations become ingrained in our perception of the world. In many cases, colorization is a simple and intuitive process for humans, as our brains are adept at making educated guesses about colors based on context and prior knowledge.
However, the task of automatic colorization, when performed by computers, serves as a prime example of how artificial intelligence can enhance our ability to extract information and meaning from grayscale images. This process can be particularly useful, even without a semantic understanding of the image. There are several key reasons why automatic colorization is valuable in various domains, including medical imaging:

1.Enhancing Visual Quality : Grayscale images are a common output from many types of medical imaging equipment. While these images provide essential diagnostic information, they may not be as visually intuitive to interpret as color images. Adding color to medical images can improve the visual quality, making it easier for healthcare professionals to identify and analyze different structures within the images.

2.Distinguishing Details : Human vision is limited in its ability to distinguish between various shades of gray in a complex medical image. Automatic colorization can help distinguish subtle differences in grayscale, making it easier to identify and analyze intricate details, such as tissues, organs, or anomalies.

3.Communication and Education : Colorization can serve as a valuable tool for communicating medical findings to patients and students. Color-coded images can help convey information more effectively, making it easier for non-experts to understand complex medical conditions and treatment plans.

While automatic colorization may not require a deep semantic understanding of the image, it relies on pattern recognition and learned associations from large datasets. Deep learning techniques, such as convolutional neural networks (CNNs), have been instrumental in automating the colorization process by learning the relationships between grayscale and color images. These models can make educated predictions about which colors should be applied to different elements within an image, based on training data.
In the field of medical imaging, automatic colorization can significantly improve the interpretability and diagnostic accuracy of grayscale images, helping healthcare professionals make more informed decisions and effectively communicate findings to patients. This technology is just one example of how AI can enhance our

ability to process and extract valuable information from visual data, even in cases where semantic understanding may not be required.

Image colorization is the process of adding color to grayscale images, making them visually appealing and realistic.

This project aims to leverage advanced machine learning techniques to automatically add color to black-and-white or grayscale images, thereby bringing them to life and evoking a sense of nostalgia and vibrancy.

This project aims to implement image colorization using the OpenCV library in Python.

OpenCV is a powerful computer vision library that provides tools and functions for image processing, making it an ideal choice for this task.

The primary objective of this image colorization project is to develop an intelligent system that can analyze and interpret grayscale images, discern the objects and scenes within them, and accurately assign appropriate colors to different elements.

Gray scale image means the value of each pixel represents only the intensity information of the light.

Such images commonly display inly the darkest black to the brightest white.

The image carries only black, white and gray colors, in which gray has multiple levels.

Each pixel typically consists of 8 bits for gray scale images and there are 256 possible grayscale colors.

2. LITERATURE SURVEY

1. “Image Colorization Based on Deep learning” - Tao Deng

   This paper presents an interactive image colorization system that combines convolutional autoencoders with user guidance to improve colorization results.

2. “Deep Learning for the Automatic Colorization of Historical Aerial Images” - Elisa Farella

   The article presents a new neural network architecture, Hyper-U-NET, which combines a U-NET-like architecture and HyperConnections to handle the colorization of historical black and white aerial images.

3. “Deep learning for image colorization: Current and future prospects” - Shanshan Huang

   In this paper, a comprehensive review of recent DLIC approaches from algorithm classifications to existing challenges is provided to facilitate researchers’ in-depth understanding of DLIC

4. "Image Colorization: A Survey and Dataset” - Saeed Anwar

   This article presents a comprehensive survey of recent state-of-the-art deep learning-based image colorization techniques, describing their fundamental block architectures, inputs, optimizers, loss functions, training protocols, and training data etc.

5. “A Survey On Auto-Image Colorization Using Deep Learning Techniques With User Proposition” - C. Santhanakrishnan

   An approach based on deep learning for automatic colorization of image with optional user-guided hints. The system maps a gray-scale image, along with, user hints” (selected colors) to an output colorization with a Convolution Neural Network (CNN).

6. “Colorization of black-and-white images using deep neural networks” - David Futschik

   Our proposed method is a fully automated process. To implement it, we propose and compare two distinct convolutional neural network architectures trained under various loss functions.

3.PROBLEM STATEMENT

This project addresses the challenge of automatically adding colors to black and white or grayscale images, enhancing their visual appeal, and potentially making them more informative for various applications such as historical image restoration, artistic image enhancement, and computer vision tasks.

4.PROPOSED SYSTEM

4.1 ALGORITHM :

For performing our project we have go through the following steps:

Step 1: Import Necessary Libraries

Step 2: Define Paths to Model Files

Step 3: Argument Parsing

Step 4: Load the Pre-Trained Model

Step 5: Configure Model for AB channel Quantization

Step 6: Load and Pre-process the input image

Step 7: Colorize the image

Step 8: Convert LAB image to BGR and Post-Processing

Step 9: Display the result

Import the required libraries, such as NumPy for numerical operations, OpenCV (cv2) for image processing, and any other libraries that may be used for argument parsing or additional functionality. Define the file paths to the pre-trained model files, including the prototxt file and the Caffe model file. These files are essential for the colorization process. To accept input parameters, use argument parsing to parse input arguments, such as the input image file name and other optional parameters. Load the colorization model using OpenCV's DNN module. This step involves using cv2.dnn.readNetFromCaffe to read the model from the prototxt and Caffe model files. Configure the model to use cluster centers for colorization. This may involve loading data related to cluster centers from a file and setting the appropriate blobs for specific layers in the colorization model. Read the input grayscale image using cv2.imread.Check if the image is loaded successfully and not empty. This step involves the core colorization process. The grayscale image is transformed into LAB color space. The L channel (lightness) of the image is used as input to the colorization model. The model performs a forward pass to generate the colorized result. Convert LAB image to BGR and Post-Processing. Finally the colorized image is displayed.

4.2 FLOWCHART :

Here is the graphical representation of the project with the help of a flowchart.

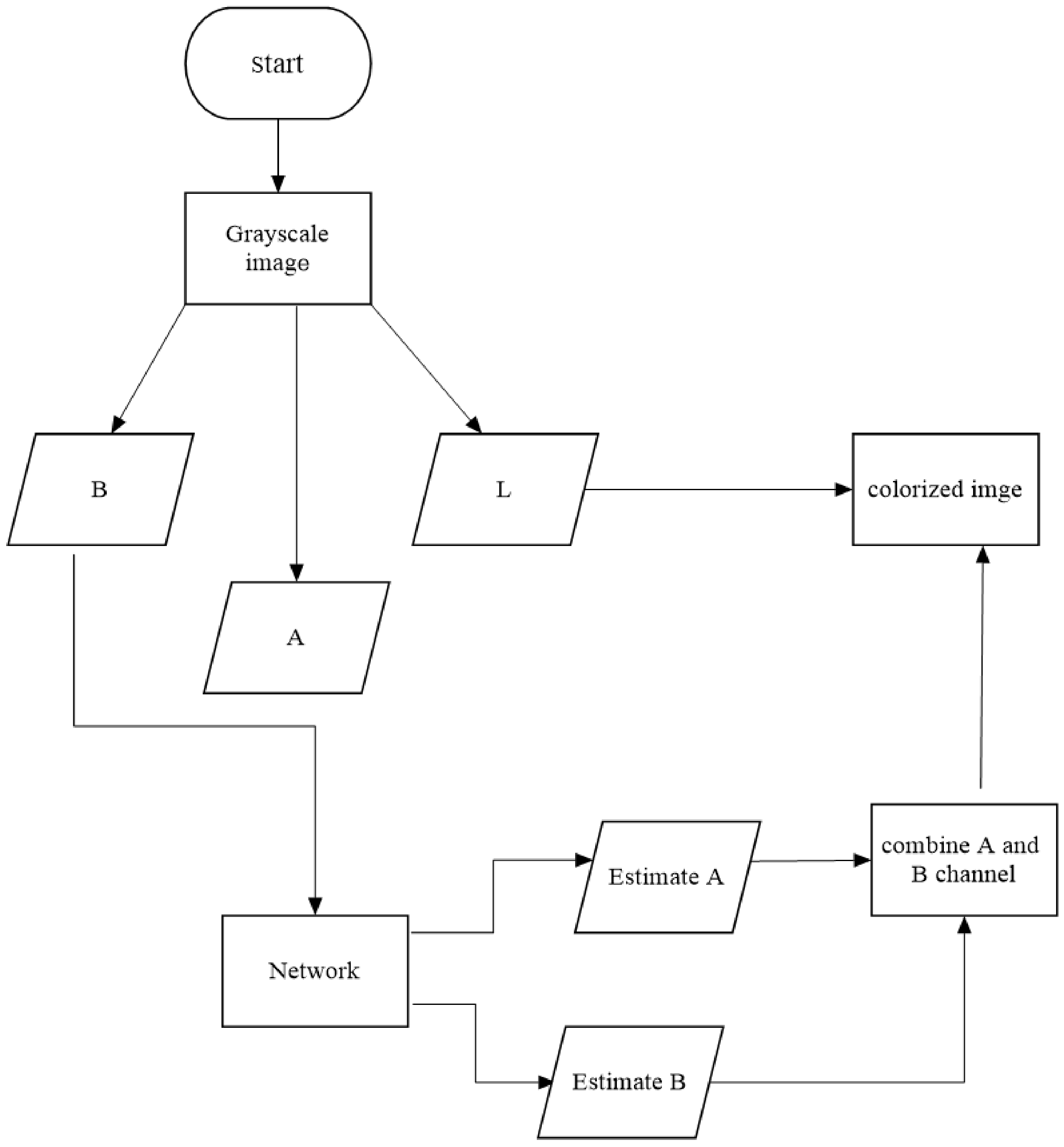


Figure 4.2.1 : Flowchart of the colorization

Take input as a grayscale image from the user . grayscale image consists of RGB values and RGB values ranges from 0 to 255 but in case of grayscale image the RGB values ranges from minimum 0 and maximum 255 i.e, RGB value 0 indicates the black color in image and RGB value 255 indicates white color in image.

We will convert the image from RGB to LAB color space where L=lightness intensity , A=red-green color combination , B=yellow-blue color combination because we want the information of the lightness(L) from the grayscale image. After conversion of RGB to LAB color space we will extract L channel from LAB color space. As we have seen earlier that L channel has lightness information. All the channel that is L,A,B is Achieved by our network or pretrained model which is based on CNN(convolution neural network).

Combining rest channels which are A and B channel. In figure 4.2.1 it can be properly seen what is missing , the only lightness is missing from our AB channel

We will combine the L(lightness intensity) channel with A(red-green color combination) and B(yellow-blue color combination) so that the Colorized image will be better but the color quality of LAB color channel is not good as RGB so we just have to convert it back to LAB to RGB which can be seen in the figure 4.2.1

4.3 BLOCK DIAGRAM

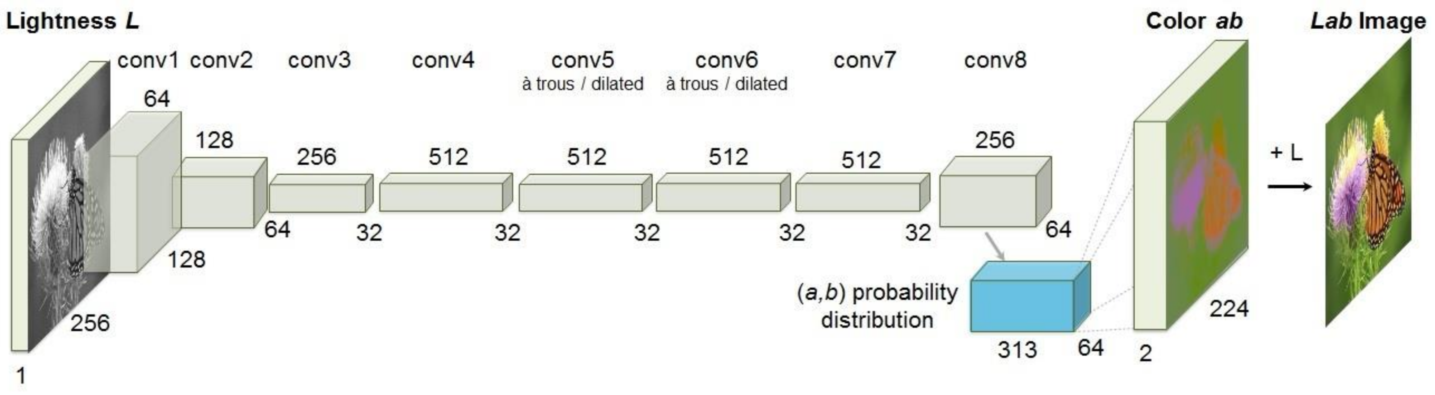
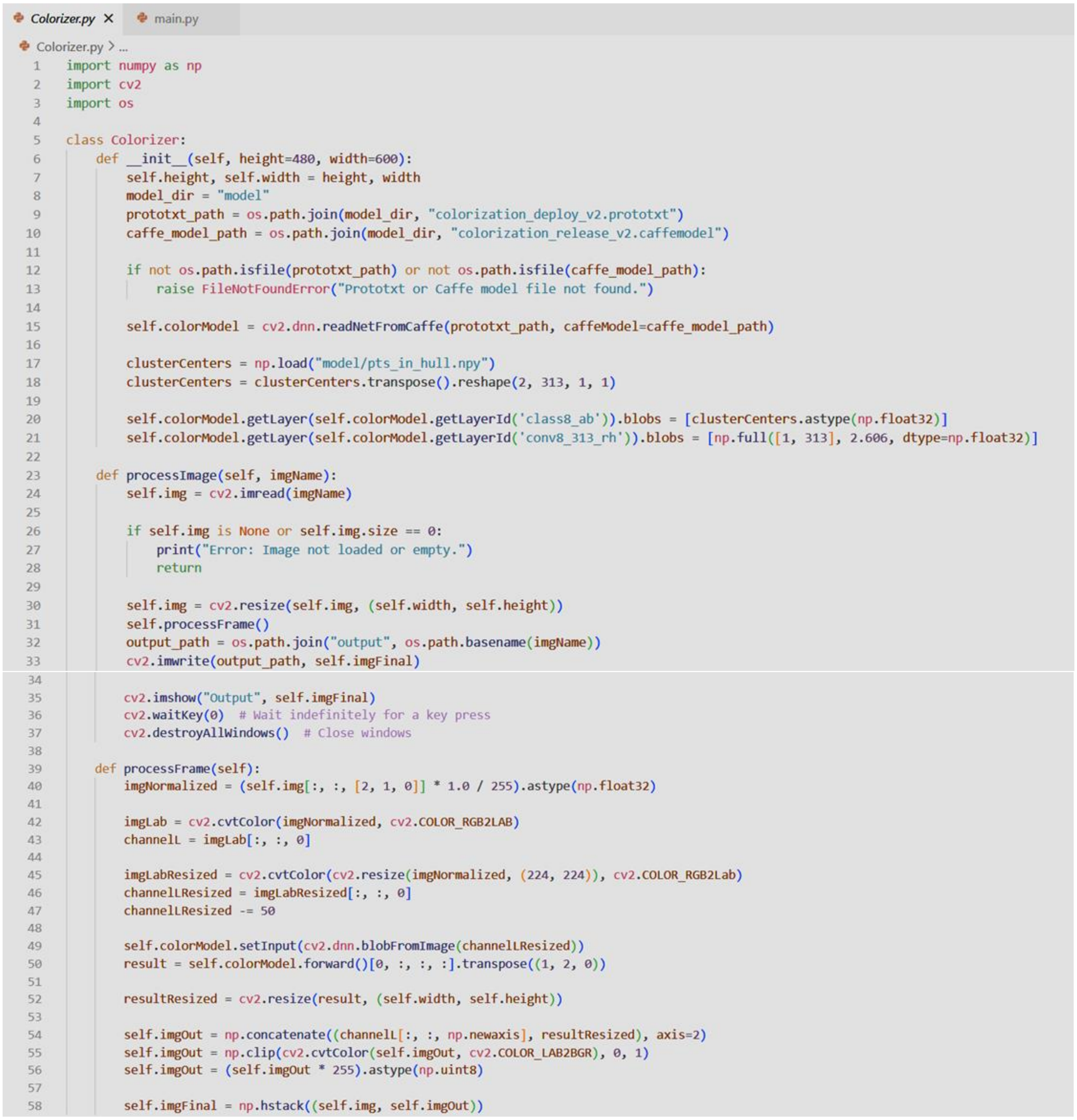


Figure 4.3.2 : Block diagram of the model for colorization with various steps

This is the black and white or grayscale image that you want to colorize. It serves as the starting point for the colorization process. Preprocessing steps may be applied to the input image to prepare it for colorization. This can include resizing, noise reduction, and enhancing image quality. The heart of the AI colorization process is a Convolutional Neural Network. CNNs are deep learning models designed for image-related tasks. In the context of colorization, a CNN is used to predict the colors for various image regions. Within the CNN, there are multiple layers responsible for feature extraction. These layers analyze the structural and textural details of the image, capturing information about edges, shapes, and patterns. The colorization module is a critical part of the CNN architecture. It takes the features extracted from the input image and generates color information for each pixel or region of the image. This module can be designed as a separate subnetwork within the overall CNN. The loss function is used to quantify the difference between the predicted colorized image and the ground. The goal is to minimize this loss during training, which helps the AI model improve its colorization accuracy. To train the AI model, a large dataset of black and white images paired with their corresponding colored versions is required. The model learns to predict colors by analyzing these pairs during training. During training, the model iteratively adjusts its parameters to minimize the loss function. This fine-tunes the model to improve its colorization capabilities. The colorized image is the final result of the AI colorization process. It is generated by applying the trained model to the input black and white image. The output image has colors added to it, making it visually appealing and vibrant.

5. CODE



```python
import numpy as np
import cv2
import os

class Colorizer:
    def __init__(self, height=480, width=600):
        self.height, self.width = height, width
        model_dir = "model"
        prototxt_path = os.path.join(model_dir, "colorization_deploy_v2.prototxt")
        caffe_model_path = os.path.join(model_dir, "colorization_release_v2.caffemodel")

        if not os.path.isfile(prototxt_path) or not os.path.isfile(caffe_model_path):
            raise FileNotFoundError("Prototxt or Caffe model file not found.")

        self.colorModel = cv2.dnn.readNetFromCaffe(prototxt_path, caffeModel=caffe_model_path)

        clusterCenters = np.load("model/pts_in_hull.npy")
        clusterCenters = clusterCenters.transpose().reshape(2, 313, 1, 1)

        self.colorModel.getLayer(self.colorModel.getLayerId('class8_ab')).blobs = [clusterCenters.astype(np.float32)]
        self.colorModel.getLayer(self.colorModel.getLayerId('conv8_313_rh')).blobs = [np.full([1, 313], 2.606, dtype=np.float32)]

    def processImage(self, imgName):
        self.img = cv2.imread(imgName)

        if self.img is None or self.img.size == 0:
            print("Error: Image not loaded or empty.")
            return

        self.img = cv2.resize(self.img, (self.width, self.height))
        self.processFrame()
        output_path = os.path.join("output", os.path.basename(imgName))
        cv2.imwrite(output_path, self.imgFinal)

        cv2.imshow("Output", self.imgFinal)
        cv2.waitKey(0)  # Wait indefinitely for a key press
        cv2.destroyAllWindows()  # Close windows

    def processFrame(self):
        imgNormalized = (self.img[:, :, [2, 1, 0]] * 1.0 / 255).astype(np.float32)

        imgLab = cv2.cvtColor(imgNormalized, cv2.COLOR_RGB2LAB)
        channelL = imgLab[:, :, 0]

        imgLabResized = cv2.cvtColor(cv2.resize(imgNormalized, (224, 224)), cv2.COLOR_RGB2Lab)
        channelLResized = imgLabResized[:, :, 0]
        channelLResized -= 50

        self.colorModel.setInput(cv2.dnn.blobFromImage(channelLResized))
        result = self.colorModel.forward()[0, :, :, :].transpose((1, 2, 0))

        resultResized = cv2.resize(result, (self.width, self.height))

        self.imgOut = np.concatenate((channelL[:, :, np.newaxis], resultResized), axis=2)
        self.imgOut = np.clip(cv2.cvtColor(self.imgOut, cv2.COLOR_LAB2BGR), 0, 1)
        self.imgOut = (self.imgOut * 255).astype(np.uint8)

        self.imgFinal = np.hstack((self.img, self.imgOut))
```

6. PROJECT OUTPUT

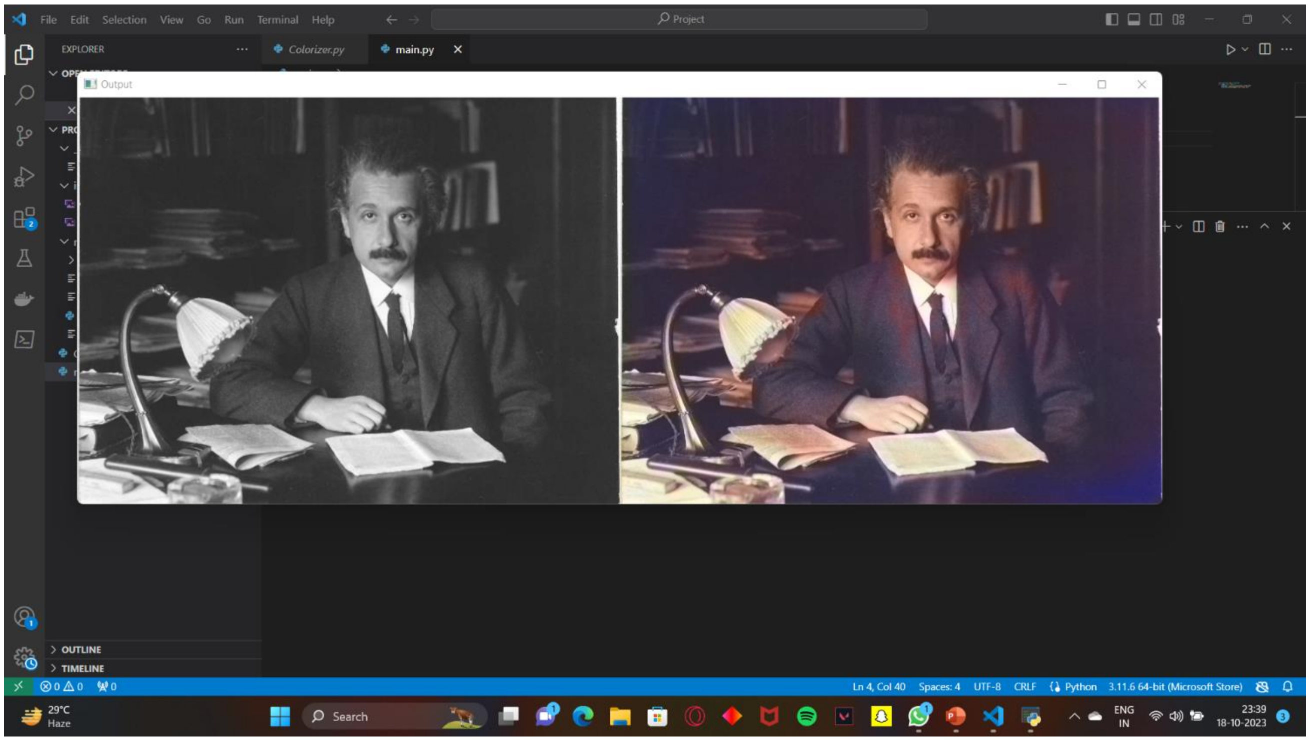

File Edit Selection View Go Run Terminal Help
Project
EXPLORER
Colorizer.py
main.py
Output
OUTLINE
TIMELINE
Ln 4, Col 40 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit (Microsoft Store)
29°C Haze
Search
ENG IN
23:39 18-10-2023

Figure 6.1 : colorization of grayscale Einstein image

In above figure 6.1 As Output image can be seen there a two images in left hand side the input image is taken as grayscale and the right hand side image is colorized image which is predicted image by the trained model . Real image and output image may differ as machine is predicting the color according to the input grayscale image. Model takes the lightness(L) information from the lab color space as we have seen in the starting.

In left hand side the image we can see that dark area that is black and the light area as white the color depth will in the output image would be based on these two values.

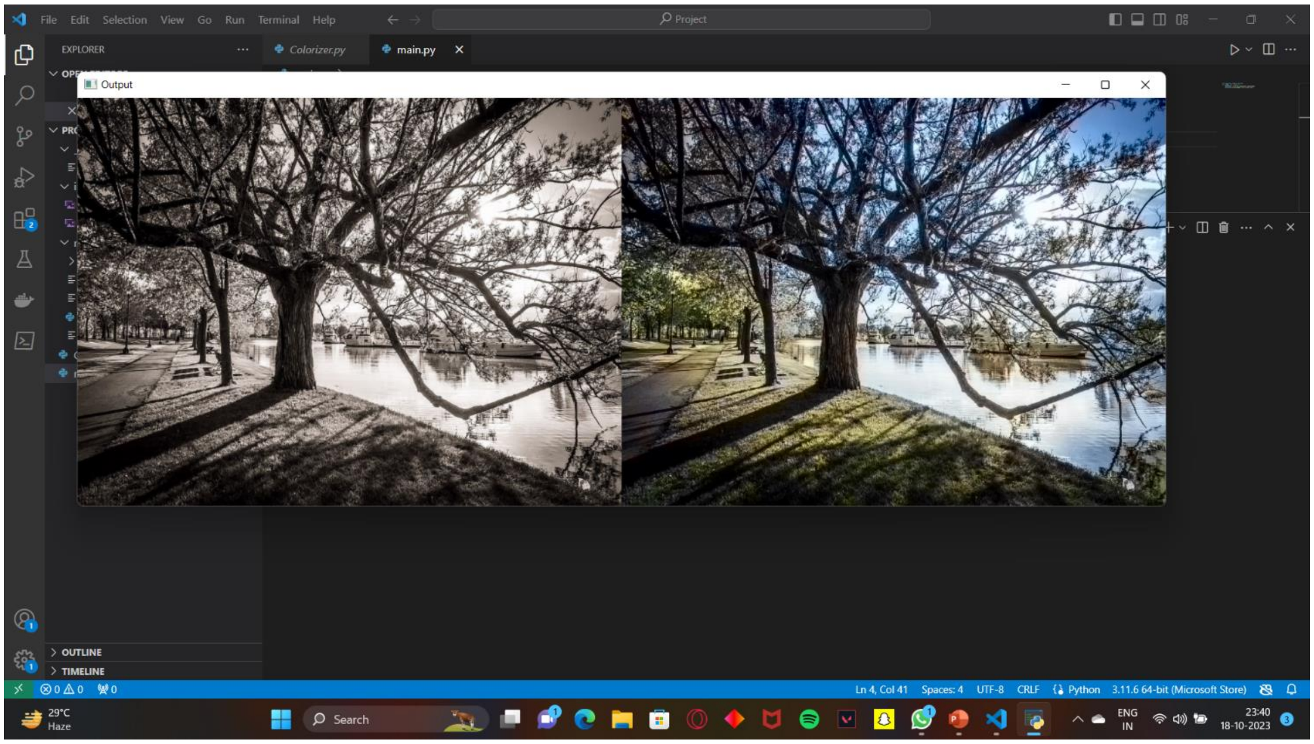


Figure 6.2 : Image Colorization of Grayscaled Lakeview Side

In figure 6.2 as we can see that the sky is blue . The model has trained on over 1000+ Images from imagenet dataset . in our childhood days we learn the things step by step. From our parents and after that we are capable of doing other things related to that. Machines also learns a pattern once they understood that things they predict things similar to that's why in above figure 6.2 model predicted that the color of the sky would be blue And that blue color also will also based on colour depth from the input image as we have seen in earlier figure 6.1.

7. CONCLUSION

We have finally completed 100% of our project and gone through various colorization sites, through many online references. We tried to develop and resolve our desired code in vs code as well as in pycharm. So after resolving all the errors, we have successfully achieved the output of our project.

8. REFERENCES

[1] “Image Colorization Based on Deep learning” - Tao Deng (2023)

[2] “Deep Learning for the Automatic Colorization of Historical Aerial Images” - Elisa Farella (2022)

[3] “Deep learning for image colorization: Current and future prospects” - Shanshan Huang (2022)

[4] "Image Colorization: A Survey and Dataset” - Saeed Anwar (2020)

[5] “A Survey On Auto-Image Colorization Using Deep Learning Techniques With User Proposition” - C. Santhanakrishnan (2019)

[6] “Colorization of black-and-white images using deep neural networks” - David Futschik (2018)