REPUBLIC OF YEMEN

SANA'A UNIVERSITY

FACULTY OF ENGINEERING

MECHATRONICS ENGINEERING DEPARTMENT

Black and White Image Colorization Using Deep Neural Networks

Prepared by:

Osama N. Al-Mesri

202170235

Abdulrahman M. Al-Aghbary

202174102

Supervised by:

Eng.Yousef

Contents

Introduction

Colorization of black and white images has long been a captivating challenge in the realm of image processing and computer vision. The process of infusing life into monochromatic images by automatically adding color has intrigued researchers and enthusiasts alike. In this project, we delve into the fascinating world of image colorization using deep neural networks, a cutting-edge technology that has revolutionized the field of artificial intelligence.

By harnessing the power of deep learning models, specifically neural networks trained on vast datasets of colored images, we aim to automate the intricate task of colorizing black and white photographs. This project stands at the intersection of art and technology, where traditional grayscale images are transformed into vivid and realistic color compositions through the intelligence of machine learning algorithms.

Through the utilization of pre-trained neural network models and sophisticated image processing techniques, we embark on a journey to breathe new life into historical black and white photographs, reviving moments from the past with a modern twist. The project not only showcases the capabilities of deep learning in creative endeavors but also underscores the seamless integration of AI-driven solutions in enhancing visual content.

Methodology

Setup and Dependencies

The script requires Python along with the OpenCV library to run. Additionally, downloading the necessary model files is crucial for the colorization process.

Neural Network Architecture

- Model Architecture: The algorithm utilizes a deep neural network architecture designed for colorization tasks. The specific details of the architecture are defined in the colorization_deploy_v2.prototxt file.

Color Quantization

- Quantization Points: The algorithm employs a set of color quantization points stored in the pts_in_hull.npy file. These points are crucial for rebalancing and quantizing the color channels for efficient colorization.

Model Loading and Initialization

- Loading Pre-Trained Weights: The pre-trained weights for the neural network model are loaded from the colorization_release_v2.caffemodel file.

Colorization Process

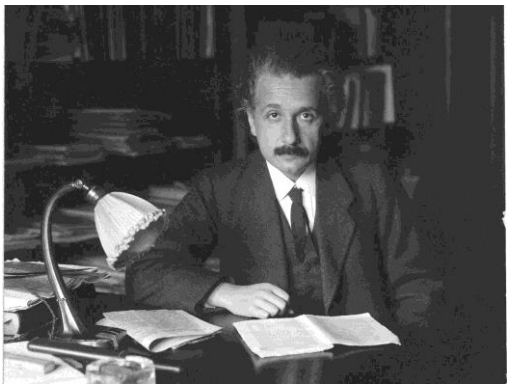
- Image Preprocessing: The algorithm first preprocesses the input black and white image. It converts the image to the LAB color space, separating the luminance (L) channel from the color channels (A and B).
- Neural Network Inference: The neural network processes the luminance channel of the image to predict the corresponding color values. This step involves passing the modified input through the network to generate color predictions.
- Color Reconstruction: After obtaining the color predictions, the algorithm reconstructs the colorized image by combining the luminance channel with the predicted A and B color channels.
- Color Space Conversion: The colorized LAB image is converted back to the RGB color space to obtain the final colorized output.

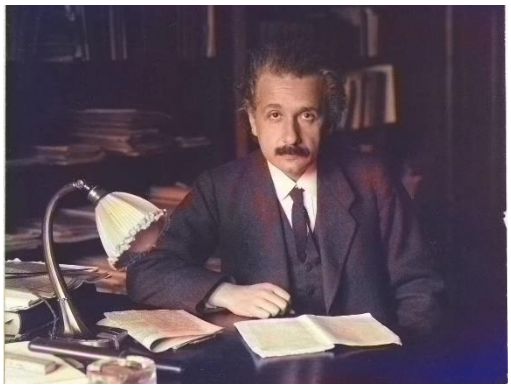
- Result Enhancement: Post-processing techniques may be applied to enhance the final colorized image quality, such as clipping pixel values and adjusting the color range.

Code along with explanation are provided in the code section.

Results

The results of some dataset I provided in the following page.





The colorization script successfully adds color to black and white images, producing visually appealing results. The neural network model demonstrates its capability to understand and predict appropriate colors for grayscale images.

Conclusion

In conclusion, the project showcases the power of deep learning in the domain of computer vision, particularly in the task of image colorization. By leveraging advanced neural network models and pre-trained weights, the script automates the process of adding color to black and white images, offering a seamless solution for enhancing visual content.

This project not only highlights the potential applications of deep learning in image processing but also serves as a practical demonstration of utilizing cutting-edge technologies to transform grayscale images into vibrant, colorful visuals.

Code

```
"""
Credits:
    1.
https://github.com/opencv/opencv/blob/master/samples/dn
n/colorization.py
    2. http://richzhang.github.io/colorization/
    3. https://github.com/richzhang/colorization/
"""

# Import necessary libraries
import numpy as np
import argparse
import cv2
import os
from os.path import split, basename, join

# Download the model files from the provided URLs

# Paths to load the model
DIR = r"C:\Users\Abdulrahman
Alghbary\Desktop\Final_Project\Colorizing-black-and-
white-images-using-Python-master"
PROTOTXT = os.path.join(DIR,
r"model/colorization_deploy_v2.prototxt")
POINTS = os.path.join(DIR, r"model/pts_in_hull.npy")
MODEL = os.path.join(DIR,
r"model/colorization_release_v2.caffemodel")
```

```
# Parse arguments using argparse
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", type=str,
required=True,
                help="path to input black and white
image")
args = vars(ap.parse_args())

# Load the Model
print("Load model")
net = cv2.dnn.readNetFromCaffe(PROTOTXT, MODEL)
pts = np.load(POINTS)

# Load centers for ab channel quantization used for
rebalancing.
class8 = net.getLayerId("class8_ab")
conv8 = net.getLayerId("conv8_313_rh")
pts = pts.transpose().reshape(2, 313, 1, 1)
net.getLayer(class8).blobs = [pts.astype("float32")]
net.getLayer(conv8).blobs = [np.full([1, 313], 2.606,
dtype="float32")]

# Load the input image
image = cv2.imread(args["image"])
scaled = image.astype("float32") / 255.0
lab = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB)

resized = cv2.resize(lab, (224, 224))
L = cv2.split(resized)[0]
```

```
L -= 50

print("Colorizing the image")
net.setInput(cv2.dnn.blobFromImage(L))
ab = net.forward()[0, :, :, :].transpose((1, 2, 0))

ab = cv2.resize(ab, (image.shape[1], image.shape[0]))

L = cv2.split(lab)[0]
colorized = np.concatenate((L[:, :, np.newaxis], ab),
axis=2)

colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
colorized = np.clip(colorized, 0, 1)

colorized = (255 * colorized).astype("uint8")


# Display the original and colorized images
cv2.imshow("Original", image)
cv2.imshow("Colorized", colorized)

# Save the colorized image
cv2.imwrite(join("output/", basename(args["image"])),
colorized)

# Wait for a key press to close windows
cv2.waitKey(0)
```

References

1. OpenCV Colorization Script
2. Rich Zhang Colorization Website
3. Rich Zhang Colorization GitHub Repository