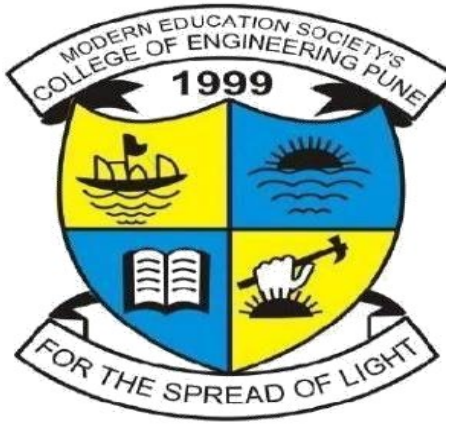
Savitribai Phule Pune University
Modern Education Society's College of Engineering, Pune, 19, Bund Garden, V.K. Joag Path, Pune – 411001.

ACCREDITED BY NAAC WITH "A++"

GRADE (CGPA – 3.13)

DEPARTMENT OF COMPUTER ENGINEERING



A REPORT ON

DL Lab: Mini Project on "Colorizing Old B&W Images: Color old black and white images to colorful images"

B.E. (COMPUTER)

SUBMITTED BY

Ms. Sejal Jadhav (PRN: F19112030)
Mr. Afnan Attar (PRN: F19112003)
Mr. Sohan Shrungare (PRN: F19112020)

UNDER THE GUIDANCE OF

Dr. (Mrs.) Shraddha Rahul Khonde

Problem Statement:
Build a deep learning model that converts old black and white images to colorful images.

Objective:
To build a deep neural network model that can Colorizing Old B&W Images: color old black and white images to colorful images using CNN.

Pre-Requisites:
- Knowledge of Python programming.
- Knowledge of Deep learning algorithms.
- Model Files:
    - colorization_deploy_v2_prototxt.
    - pts_in_hull.npy.
    - colorization_release_v2.caffemodel.

Theory:

Colorizing black and white images using deep learning models is a task that has been extensively researched in recent years. Deep learning models use the available color images to learn the relationship between the black and white and color images. Once trained, these models can colorize any black and white image by predicting the color channels from the brightness channel. There are various deep learning models that can be used for image colorization, including Generative Adversarial Networks (GANs), Convolutional Neural Networks (CNNs), and Autoencoders.

In this project, a pre-trained deep learning model is used to colorize a black and white image. The model used in this code is a Convolutional Neural Network (CNN), which has been trained on a large dataset of color images. The model predicts the AB channels of the LAB color space from the L channel, where the L channel represents the brightness information of the image, and the AB channels represent the color information.

The pre-trained model is loaded into memory, and the required files are read from the disk. The centers for the AB channel are loaded, which were obtained by clustering the AB channels of the color images in the training dataset. These centers are then used to colorize the input image.

The input image is converted to the LAB color space, and the brightness information is extracted from the L channel. The brightness information is then passed to the pre-trained model, which predicts the AB channels. The predicted AB channels are then concatenated with the L channel to obtain the colorized image. Finally, the colorized image is converted back to the BGR color space and displayed to the user.

Process:

First, the required libraries are imported, including NumPy, argparse, OpenCV (cv2), and os.

Then, the code loads the pre-trained model files from the specified directory and sets up an argument parser to allow the user to provide the path to the input black and white image.

After that, the code loads the pre-trained model, followed by loading the centers for the AB channel. The AB channel is the color information of the image, and the pre-trained model uses a clustering technique to determine the centers of the AB channel. These centers are then used to colorize the input image.

Next, the code loads the input image, converts it to the LAB color space, and resizes it to 224x224 pixels. The L channel, which represents the brightness information, is then subtracted by 50 to reduce the brightness.

The code then uses the pre-trained model to colorize the input image by passing the L channel to the model, obtaining the predicted AB channels, and concatenating them with the L channel to obtain the colorized image. The colorized image is then converted back to the BGR color space, clipped between 0 and 1, and multiplied by 255 to obtain the final colorized image.

Finally, the code displays the original and colorized images using OpenCV's imshow function and waits for the user to press a key to close the windows.

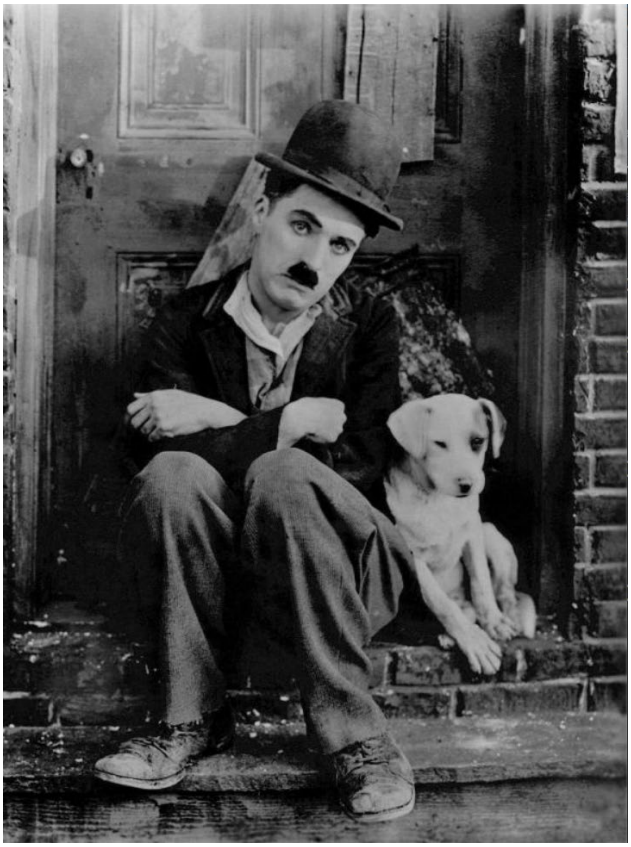
In summary, this code performs image colorization using a pre-trained deep learning model, demonstrating the use of libraries like NumPy, argparse, and OpenCV to load and process the image and model files.

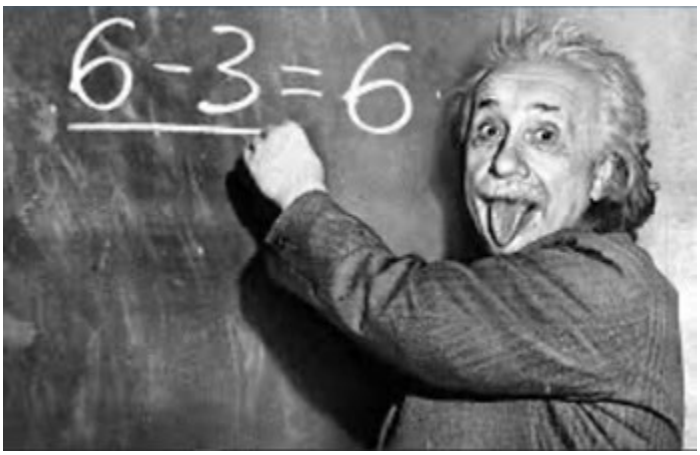
Code:

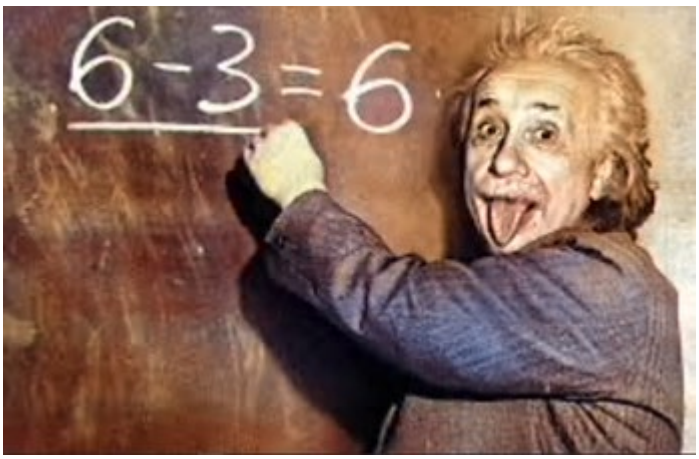
```
# Import Modules
import numpy as np
import argparse
import cv2
import os

'''
 Training Dataset is computationally expensive and dataset required is
quite large.
 Hence we will use Pre-trained Models.
 Model files we used:
 1. colorization_deploy_v2_prototxt.
 2. pts_in_hull.npy.
 3. colorization_release_v2.caffemodel
[https://www.dropbox.com/s/dx0qvhhp5hbcx7z/colorization_release_v2.caffem
odel?dl=1]

'''


# Load Files
DIR = r"./colorize"
PROTOTXT = os.path.join(DIR, r"colorization_deploy_v2.prototxt")
POINTS = os.path.join(DIR, r"pts_in_hull.npy")
MODEL = os.path.join(DIR, r"colorization_release_v2.caffemodel")

# Argparser
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", type = str, required = True, help =
"Path to input BW image")
args = vars(ap.parse_args())

# Loading the model
print("Load Model...")
```

```
net = cv2.dnn.readNetFromCaffe(PROTOTXT, MODEL)
pts = np.load(POINTS)

# Loading Centres for AB channel
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")]

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

# Tune parameters
resized = cv2.resize(lab, (224, 224))
L = cv2.split(resized)[0]
L -= 50

# Colorizing
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 Output
cv2.imshow("Original", image)
cv2.imshow("Colorized", colorized)
cv2.waitKey(0)
```

Output:







Conclusion:

Thus, Black and White images were converted to color images using CNN.