Image Colorization Using Deep Learning

MINI-PROJECT REPORT

Submitted for

CSE2009 – Soft Computing (Slot-D)

by

Sangam Mahajan (18BCE7005)

SCHOOL OF COMPUTER SCIENCE ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

ABSTRACT

Sometimes technology enhances art. Sometimes it vandalizes art. Colorizing black and white films is a very old idea dating back to 1902. For decades many movie creators opposed the idea of colorizing their black and white movies and thought of it as vandalism of their art. Today it is accepted as an enhancement to the art form.

The technology itself has moved from painstaking hand colorization to today's largely automated techniques. In the United States, Legend Films used its automated technology to color old classics. In India, the movie Mughal-e-Azam, a blockbuster released in 1960 was remastered in color in 2004. People from multiple generations crowded the theaters to see it in color and the movie was a huge hit for the second time!

TABLE OF CONTENTS

- **INTRODUCTION**

In the last few years, with the Deep Learning revolution, automation in colorization has taken a huge leap forward. In this post, we will learn about one such Deep Learning model. We also share OpenCV code to use the trained model in a Python or C++ application.

In ECCV 2016, Richard Zhang, Phillip Isola, and Alexei A. Efros published a paper titled Colorful Image Colorization in which they presented a Convolutional Neural Network for colorizing gray images. They trained the network with 1.3M images from the ImageNet training set. The authors have also made a trained Caffe-based model publicly available. In this post, we will first define the colorization problem, explain the architectural details of the paper and finally share the code and some fascinating results.

Many colorization papers have been published using traditional computer vision methods. One of my favorites is a paper titled Colorization using Optimization by Anat Levin, Dani Lischinski, and Yair Weiss. It used a few colored scribbles to guide an optimization problem for solving colorization.

- **Literature Review**

Image colorization is the process of taking an input grayscale (black and white) image and then producing an output colorized image that represents the semantic colors and tones of the input (for example, an ocean on a clear sunny day must be plausibly “blue” — it can’t be colored “hot pink” by the model).

Previous methods for image colorization either:

- Relied on significant human interaction and annotation
- Produced desaturated colorization

The novel approach we are going to use here today instead relies on deep learning. We will utilize a Convolutional Neural Network capable of colorizing black and white images with results that can even “fool” humans!

- **Problem Definition**

Let's first define the colorization problem in terms of the CIE Lab color space. Like the RGB color space, it is a 3-channel color space, but unlike the RGB color space, color information is encoded only in the a (green-red component) and b (blue-yellow component) channels. The L (lightness) channel encodes intensity information only.

The grayscale image we want to color can be thought as the L-channel of the image in the Lab color space and our objective is to find the a and b components. The Lab image so obtained can be transformed to the RGB color space using standard color space transforms. For example, in OpenCV, this can be achieved using cvtColor with COLOR_BGR2Lab option.

To simplify calculations, the ab space of the Lab color space is quantized into 313 bins. Instead of finding the a and b values for every pixel, because of this quantization, we simply need to find a bin number between 0 and 312. Yet another way of thinking about the problem is that we already have the L channel that takes values from 0 to 255, and we need to find the ab channel that takes values between 0 to 312. So the color prediction task is now turned into a multinomial classification problem where for every gray pixel there are 313 classes to choose from.

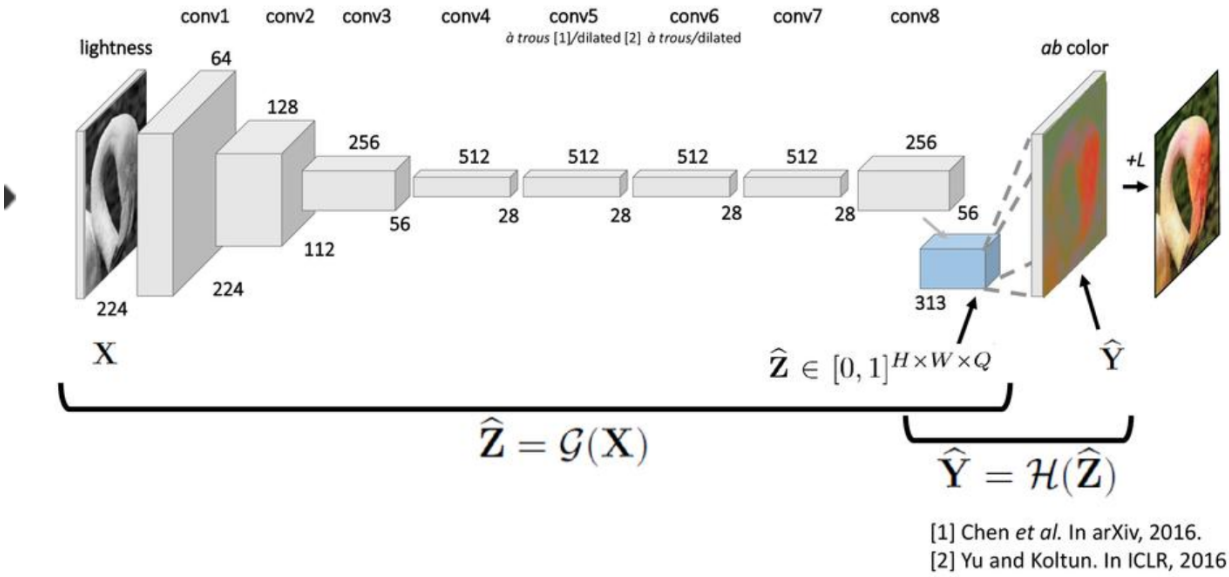
You may be tempted to think that we can simply take the mean of the distribution and choose the ab pair corresponding to the nearest quantized bin center. Unfortunately, this distribution is not Gaussian, and the mean of the distribution simply corresponds to an unnatural desaturated color. To understand this, think of the color of the sky — it is sometimes blue and sometimes orange-yellow. The distribution of colors of the sky is bi-modal. While coloring the sky, either blue or yellow will result in a plausible coloring. But the average of blue and yellow is an uninteresting gray.

- **Implementation**

4.1 Hardware Details

CNN Architecture for Colorization

The architecture proposed by Zhang et al is a VGG-style network with multiple convolutional blocks. Each block has two or three convolutional layers followed by a Rectified Linear Unit (ReLU) and terminating in a Batch Normalization layer. Unlike the VGG net, there are no pooling or fully connected layers.



The input image is rescaled to 224×224. Let us represent this rescaled grayscale input image by X. When it passes through the neural network shown above, it gets transformed to $\hat{Z}$ by the neural network. Mathematically, this transformation G by the network can be written as $\hat{Z} = G(X)$

The dimensions of $\hat{Z}$ is $H \text{ x } W \text{ x } Q$, where $H(= 56)$ and $W(= 56)$ are the height and width of the output of the last convolution layer. For each of the $H \text{ x } W$ pixels, $\hat{Z}$ contains a

vector of $Q(=313)$ values where each value represents the probability of the pixel belonging to that class. Our goal is to find a single pair of ab channel values for each probability distribution $\hat{Z}_{h,w}$.

Recovering color image from $\hat{Z}$

The CNN shown in Figure 3 gives us a collection of distributions in $\hat{Z}$ from the resized input image X. Let's see how to recover a single ab value pair from each distribution in $\hat{Z}$.

You may be tempted to think that we can simply take the mean of the distribution and choose the ab pair corresponding to the nearest quantized bin center. Unfortunately, this distribution is not Gaussian, and the mean of the distribution simply corresponds to an unnatural desaturated color. To understand this, think of the color of the sky — it is sometimes blue and sometimes orange-yellow. The distribution of colors of the sky is bi-modal. While coloring the sky, either blue or yellow will result in a plausible coloring. But the average of blue and yellow is an uninteresting gray.

So why not use the mode of the distribution so you get either blue or yellow sky? Sure, the authors tried that and while it gave vibrant colors, it sometimes broke spatial consistency. Their solution was to interpolate between the mean and mode estimates to obtain a quantity called the annealed-mean. A parameter called temperature (T) was used to control the degree of interpolation. A final value of T=0.38 is used as a trade-off between the two extremes.

Notice that when the image passes through the CNN, its size decreases to 56×56. Therefore the predicted ab image, $\hat{Y}$, also has the dimension 56×56. To obtain the color image, it is upsampled to the original image size and then added to the lightness channel, L, to produce the final color image.

4.2 Software Details

Color Rebalancing

To nudge the algorithm to produce vibrant colors, the authors changed the loss function to

$$L(\hat{Z}, Z) = -\frac{1}{HW} \sum_{h,w} v(Z_{h,w}) \sum_{q} Z_{h,w,q} \log(\hat{Z}_{h,w,q})$$

The color rebalancing term, $v(.)$ is used to rebalance the loss based on the rarity of the color class. This contributes towards getting more vibrant and saturated colors in the output.

Results of Colorization

With and without color rebalancing. We tried both versions and have shared the results in the Figure below. The middle column shows the version without color rebalancing and the last column shows the version with rebalancing.

As we can see, color rebalancing makes many images very lively and vibrant. Most of them are plausible colors. On the other hand, sometimes it could also add some unwanted saturated color patches to some images.

Keep in mind that when we try to convert a grayscale image into a color image, there could be multiple plausible solutions. So the way to evaluate good colorization is not how well it matches the ground truth, but how plausible and pleasant it looks to the human eye.

Animals

The model does very well on images of animals — especially cats and dogs. This is because ImageNet contains a very large collection of these animals.

- **Read model**

Here, we provide the paths to the protoFile and the weightsFile in the code. By selecting the appropriate model, depending on whether to use color rebalancing or not. Here default it to using the color rebalancing model. It reads the input image and defines the network's input size to be 224×224.

- **Load quantized bin centers**

Next, we load quantized bin centers. We then assign 1×1 kernels corresponding to each of the 313 bin centers and assign them to the corresponding layer in the network. Finally, we add a scaling layer with a non-zero value.

- **Convert image to CIE Lab color space**

The input RGB image is scaled so that the values are in the range 0-1, and then it is converted to Lab color space and the lightness channel is extracted out. The lightness channel in the original image is resized to the network input size which is (224,224) in this case. Usually, the lightness channel ranges from 0 to 100. So we subtract 50 to center it at 0.

Then we feed the scaled and mean centered lightness channel to the network as its input for the forward pass. The output of the forward pass is the predicted ab channel for the image. It is scaled back to the original image size and then merged with the original sized lightness image(extracted earlier in original resolution) to get the output Lab image. It is then converted to RGB color space to get the final color image. We can then save the output image.

4.3 Screenshots

- **Input**

- **Output**

4.4 Sample Code

b2w.py

```
import numpy as np

import cv2


print("loading models.....")


protoFile = "./models/colorization_deploy_v2.prototxt"

weightsFile = "./models/colorization_release_v2.caffemodel"
```

```
net = cv2.dnn.readNetFromCaffe(protoFile,weightsFile)
pts = np.load('./models/pts_in_hull.npy')



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')]



image = cv2.imread('./image/image1.jpg')
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



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")


cv2.imshow("Original",image)

cv2.imshow("Colorized",colorized)


cv2.waitKey(0)
```

- **Results and Discussion**

I’m particularly impressed by this image colorization.

Previous approaches to black and white image colorization relied on manual human annotation and often produced desaturated results that were not “believable” as true colorizations. I have also used Opencv to read the image but Opencv reads the image in BGR format, so we need to convert the image to RGB format first before turning it to grayscale.

- **Conclusions & Scope for future**

In the following images we can see that 1. b/w to its 3. original is different from the 2. colorized pictures using deep learning.

Image keeps on changing its color whenever we take outputs again and again, the whole process works again and always gives fresh output.



The best way to find the quality of an image is to look at it because human eyes are the ultimate viewer. Subjective image quality is concerned with how an image is perceived by a viewer and gives his or her opinion on a particular image. The 'Mean Opinion Score' (MOS) can be used for subjective quality assessment. In standard subjective tests where the observers view image and assess its quality by assigning to it a category of a given rating scale. In comparative evaluation, a set of images are ranked from best to worst by the observers.

7. References

- Colorful Image Colorization, Richard Zhang, Phillip Isola, Alexei A. Efros, ECCV 2016
- OpenCV Samples on Colorization
- Images and videos used in this post are in the public domain and were obtained from the following sources: Wikimedia Commons, Max Pixel, pxhere, Pixabay [1], [2], [3], [4], [5] and Pexels