Implement of Colorful Image Colorization

Yanbaihui Liu
University of Michigan
yanbhliu@umich.edu

Pengfei Gao
University of Michigan
pfgao@umich.edu

Yiming Li
University of Michigan
billyli@umich.edu

Abstract

Due to the uncertainty, automatic colorization is always a challenging task for people. Nowadays, it has become feasible to use Convolutional Neural Network (CNN) to colorize grayscale images efficiently and effectively. In this project, we used CNN to predict the AB channel of images according to the given L channel (which represents grayscale images). To get a better result, we used cross-entropy as the loss function instead of using MSE loss function. Compared with the traditional L2 norm method, the color reproduction by our method seems more realistic. Although the restricted calculation resource limits the performance of our model, the result still shows that this method is feasible and meaningful.

1. Introduction

In the past, due to the limitation of technology, people could only take grayscale photographs. Because different kinds of color can be transformed to the same grayscale pixel, this kind of photograph lacks a lot of detail, which limits research in many different fields.

Before machine learning came along, people used traditional methods to colorize photos, which could not work very well. Therefore, using machine learning to colorize images is an effective way to solve this problem. In this project, we introduced the colorization task as an instance of cross-channel encoding. We developed a Convolutional Neural Network(CNN) to train a model which could realize the statistical dependencies between the semantics and the textures of grayscale images and their color versions. This project is inspired by the paper "Colorful Image Colorization" by Richard Zhang, etc. [5].

Due to the limitation of calculation ability of GPU and time, we reduced the shape of the train set. In addition, to get a better performance in a limited time, we changed the architecture of the network. According to our results, the final loss after 20 epochs is 14.053 and the color reproduction result is meaningful.

2. Related Work

Before machine learning became popular, image colorization mainly need users' guides (Scribble-based colorization) or color reference images(Example-based colorization). Levin et al. proposed a method to colorize images that need users to annotate the image with color scribbles [3]. This method is based on the fact that neighboring pixels with similar luminance also have similar colors. However, this method depends on the color users choose. Thus, users have to consider the color for every region carefully. After several years, Chia et al. presented a method that colorizes images based on reference images [2]. For this method, an unsuitable reference would cause terrible results. Besides that, one reference image can not have effects on various kinds of images.

With the development of machine learning and deep learning, finding similar image patches/pixels for colorization based on a large amount of data has become the most popular way to colorize. Using a convolutional neural network to predict color can be thought of as an encoder/decoder system. Thus, to evaluate the performance of coloring, defining a suitable loss function is vital. Cheng et al. chose mean squared error(MSE) function as the loss function to implement colorization [1]. However, MSE function always encourages conservative predictions, thus, the colorization output can not show bright color. In 2016, Richard Zhang et al. proposed a new loss function based on the cross-entropy loss function. This loss function introduced class rebalancing to encourage the system to learn more from rare colors. Consequently, the result of this method showed a more effective performance. Although some more effective methods [4] were proposed in the recent two years, Richard Zhang's is still representative. Therefore, we decide to develop our project based on this paper.

3. Method

To colorize images, images need to be converted to LAB color space, which a channel contains green-red components, b channel contains the blue-yellow component, and

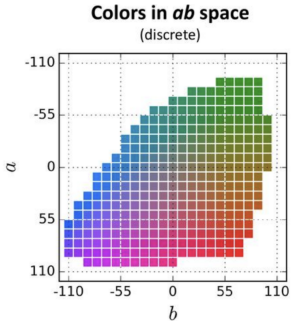

Figure 1. Quantized colors in ab channel.

L channel encodes the intensity information. The basic idea behind the scene is to measure the divergence between L and the objective of a,b channels. However, simply using the L2 norm between the real image and the generated image is not robust to the inherent ambiguity and multi-model of the coloring problem and may average out colors and finally give a greyish result. That's why we decided to adopt Zhang et al.'s work to makes the colorization problem a multinomial classification problem by first divided the whole LAB color space into 313 bins of size 10 shown in Figure 1; instead of finding a,b for every pixel, we find a bin between 0 to 312, which is like HOG that we have done in PS3. We then set up a ConvNet to transform the lightness of a given image X into $\hat{Z} \in [0,1]^{H \times W \times Q}$, so for each $H \times W$ pixels, $\hat{Z}$ contains the probability that a pixel belongs to that bin. Due to the limitation of time and GPU resources, we only trained 10000 images, and the architecture of the network is shown in Figure 2. To compare the predicted result and real data, we need to define a function $Z = \mathcal{H}^{-1}(Y)$ that can convert real color Y to its corresponding Z value. Specifically, this will trans Y into a One-Hot vector by searching for the closest bin. For better result, 5-nearest neighbors are grouped together, and Gaussian distribution is adapted to find Z based on the distance to the ground truth Y. Finally, the Cross-Entropy loss defined as

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

where h and w are height and width of images, and q is bin number, and v is used to rebalance the loss based on the rarity of the color class. $\hat{Y}$, which mapped by the predicted probability disribution $\hat{Z}$, is upsampled from 56×56 to the original size and added back to L to form the final predicted colorful iamge.

4. Experiments

Considering the computational limit of this project, we train our network on 2000 images from a subset of Intel Image Classification Dataset, including glaciers and mountains, and test on a few separate images in the validation set. The training and testing images are in size of 150×150. We chose our batch size, and number of epochs to be 32 and 30. We also use a self-adaptive learning rate that starts from 0.001 and decays to $\frac{1}{10}$ of the previous every 5 epochs. For color rebalancing, in another word, mitigating the biased distribution typically caused by image backgrounds, we chose the temperature parameter to be 0.38 as [5] suggests.

Layer (type)	Output Shape
Conv2d-1	[-1, 64, 256, 256]
ReLU-2	[-1, 64, 256, 256]
Conv2d-3	[-1, 64, 128, 128]
ReLU-4	[-1, 64, 128, 128]
BatchNorm2d-5	[-1, 64, 128, 128]
Conv2d-6	[-1, 128, 128, 128]
ReLU-7	[-1, 128, 128, 128]
Conv2d-8	[-1, 128, 64, 64]
ReLU-9	[-1, 128, 64, 64]
BatchNorm2d-10	[-1, 128, 64, 64]
Conv2d-11	[-1, 256, 64, 64]
ReLU-12	[-1, 256, 64, 64]
Conv2d-13	[-1, 256, 64, 64]
ReLU-14	[-1, 256, 64, 64]
Conv2d-15	[-1, 256, 32, 32]
ReLU-16	[-1, 256, 32, 32]
BatchNorm2d-17	[-1, 256, 32, 32]
Conv2d-18	[-1, 512, 32, 32]
ReLU-19	[-1, 512, 32, 32]
Conv2d-20	[-1, 512, 32, 32]
ReLU-21	[-1, 512, 32, 32]
Conv2d-22	[-1, 512, 32, 32]
ReLU-23	[-1, 512, 32, 32]
BatchNorm2d-24	[-1, 512, 32, 32]
Conv2d-25	[-1, 512, 32, 32]
ReLU-26	[-1, 512, 32, 32]
Conv2d-27	[-1, 512, 32, 32]
ReLU-28	[-1, 512, 32, 32]
Conv2d-29	[-1, 512, 32, 32]
ReLU-30	[-1, 512, 32, 32]
BatchNorm2d-31	[-1, 512, 32, 32]
Conv2d-32	[-1, 512, 32, 32]
ReLU-33	[-1, 512, 32, 32]
Conv2d-34	[-1, 512, 32, 32]
ReLU-35	[-1, 512, 32, 32]
Conv2d-36	[-1, 512, 32, 32]
ReLU-37	[-1, 512, 32, 32]
BatchNorm2d-38	[-1, 512, 32, 32]
Conv2d-39	[-1, 256, 32, 32]
ReLU-40	[-1, 256, 32, 32]
Conv2d-41	[-1, 256, 32, 32]
ReLU-42	[-1, 256, 32, 32]
Conv2d-43	[-1, 256, 32, 32]
ReLU-44	[-1, 256, 32, 32]
BatchNorm2d-45	[-1, 256, 32, 32]
ConvTranspose2d-46	[-1, 128, 64, 64]
ReLU-47	[-1, 128, 64, 64]
Conv2d-48	[-1, 128, 64, 64]
ReLU-49	[-1, 128, 64, 64]
Conv2d-50	[-1, 128, 64, 64]
ReLU-51	[-1, 128, 64, 64]
Conv2d-52	[-1, 313, 64, 64]

Figure 2. Network Architecture.

Table 1. Comparison Metrics

	Baseline	Our Good	Our Bad
psnr	21.325	26.663	21.688
ssim	0.846	0.959	0.925

Figure 3. The Predicted Result Using Simple MSE And Its Original Image.

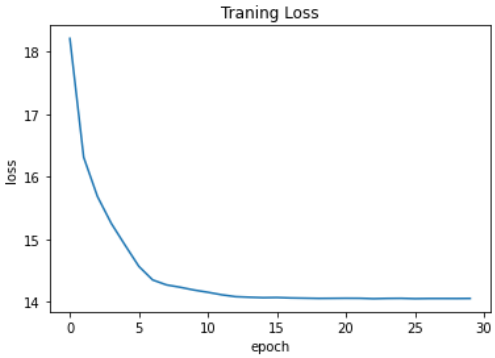

Figure 4. Training Loss.

Figure 4 shows the trends of our training loss. Since we use a small dataset with only two categories, the objective function converge as soon as approximately 20 epochs. Our colorization testing results is shown in Figure 5. Some good generating results are shown in Figure 5, and some bad results are shown in Figure 6. Comparing the real image, in most cases, colorized images looks real but with low saturation, especially for color other than blue. This is because in our training set, icebergs and glaciers and some mountains with large areas of blue sky take the main part, so our model does not learn other colors very well.

We use the model with MSE loss function as the baseline model and choose an image that gives good results with our method as the test image, and the result is shown in Figure 3. Intuitively, the results generated from our model are more realistic. To scientifically measure the colorization quality, we adopt PSNR and SSIM as our metrics, as shown in Table1. PSNR and SSIM calculated from our method are the average value of three images pairs. By comparing the data in the three columns, we can not only visualize that the MSE results are indecent and even much worse than our bad results as shown in graph, but also prove it numerically.

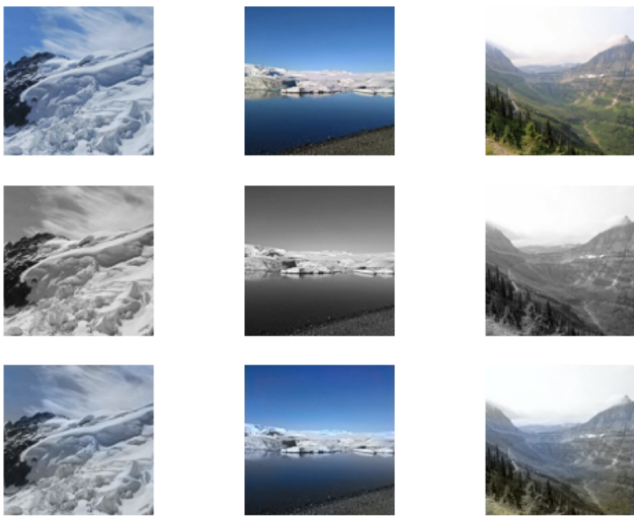

Figure 5. Good Testing Results. First Row Are Given Images, Second Row Are Greyscale Images, and Thirt Row Are Predicted Images.

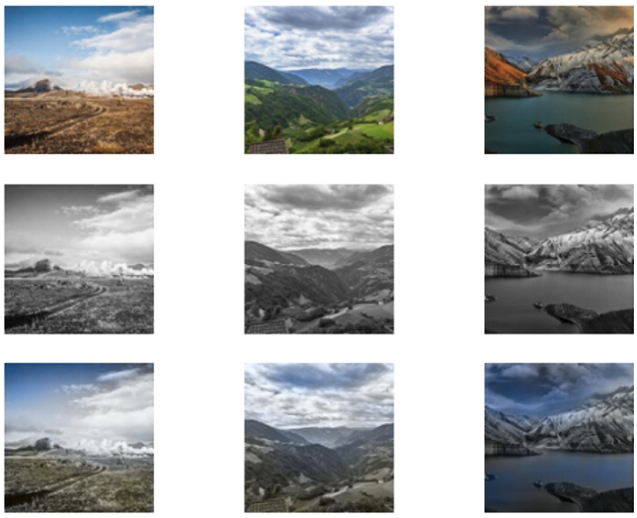

Figure 6. Bad Testing Results. First Row Are Given Images, Second Row Are Greyscale Images, and Thirt Row Are Predicted Images.

5. Conclusions

Overall, our method uses the LAB color space and transforms the colorization problem into a classification problem. Compared with the traditional L2 norm method of calculating loss, the color reproduction by our method is more realistic. However, due to the limitation of resource and time, compared with the papers we refer to, we have greatly reduced the training set and epoch, which makes the result not ideal. We expected to see improvement if we continuously train it with larger dataset that contains more categories.

References

[1] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[2] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. Image colorization using similar images. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 369–378, 2012.

[3] LISCHINSKI-D. LEVIN, A. and Y. WEISS. Colorization using optimization. In *ACM Trans. Graph. (Proc. of SIGGRAPH)*, pages 689—694, 2004.

[4] Tien-Tsin Wong Yi Ji Lvmin Zhang, Chengze Li and Chunping Liu. Two-stage sketch colorization. In *ACM TOG (Proc. SIGGRAPH Asia)*, page 261:1–261:14, 2018.

[5] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.