Image Colorization using Regression, Classification and GAN

Weiji Li
University of Michigan
weijili@umich.edu

Haoyi Qiu
University of Michigan
haoyiqiu@umich.edu

Yizhi Tang
University of Michigan
tangyz@umich.edu

Lingzi Liao
University of Michigan
lynliao@umich.edu

Zilin Zhang
University of Michigan
zzilin@umich.edu

1. Introduction

1.1. Motivation

Classic old photos could be limited by the technology of the era and passed down only in the grayscale form. Although black and white photos can sometimes reveal special poetic feelings, colorized photos give people a stronger sense of immersion. Restoring black and white photos is a popular and interesting topic in society. Our group tries to explore ways to do the restoration to colorize the grayscale input images with plausible color versions. Instead of training models to restore the original colors, we aim to achieve the colorization that is considered reasonable by people.

We learned about the basic concepts and implementations of neural networks. Convolutional neural networks (CNN) are popular models in the image colorization field based on its strong ability of learning the features at various levels through convolution and pooling operations. For image processing, CNNs examine features of different objects on images from helpful aspects for image colorization, including color, brightness, and local details such as edges, corners, and lines. In our project, we choose to develop our models using convolutional neural networks.

1.2. Related Work

Our models are mainly inspired by the ideas in [9] and [6]. Our regression and classification models use CNNs to extract features from the images and predict the potential colors by regression and color distribution respectively. Our generative adversarial networks (GAN) [3] models use the idea of generator and discriminator from [6] and we also adapted its network structure.

1.3. Summary

Our models take L (perceptual lightness) channel in LAB color space as input and predict the AB (four unique colors of human vision: red, green, blue, and yellow) channels after feeding large-scale training data of colorful images. We implement regression, multinomial classification, and GAN in this project, and evaluate the colorization results using quantitative evaluation and human study methods, respectively, to compare the performance between the three models.

2. Data

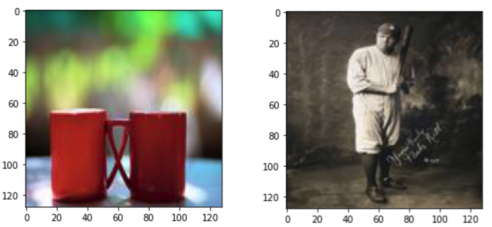

Figure 1. A demonstration of images and their colorfulness score.

The dataset is chosen from colorful images in ImageNet-ILSVRC2014 [1] Dataset. We randomly pick about 20,000 images from 569 object categories and adjust the resolution to 128×128 pixels in LAB color space. The training, testing, and validation split varies between models and we will discuss it in the Experiment section. Within the randomly selected images, some are not as colorful. There is little difference between the original images and the converted grayscale images and thus they are not helpful for training purposes. To solve this problem, as demonstrated in the figure 1, we apply the Hasler and Süsstrunk's approach [4], calculating a colorfulness score for each image and picking out colorful images. Specifically, among 16000 training images, we choose the top 11849 colorful images as our training dataset.

[1] http://www.image-net.org/challenges/LSVRC/2014/

3. Approaches

Recently, deep neural networks have shown remarkable success in image colorization. This success may be due to their ability to capture and use semantic and textural information in the original image ([2], [2]). Therefore, our work leverages large-scale data and relies on fully convolutional neural networks.

We work with the images in the LAB colorspace, which contains the same information as RGB, but is easier to separate the lightness channel from the other two. The grayscale images can be thought of as the L-channel of the images in the LAB colorspace, and our objective is to find the A and B components.

We aim to produce a colored image with three channels per pixel from a grayscale image with only one channel per pixel (lightness). For simplicity, we will only work with images of size 128×128. Therefore, our inputs are of size $1 \times 128 \times 128$ (the lightness channel of original colored images), and outputs are of size $2 \times 128 \times 128$ (the predicted two channels). We try to predict the color values of the input grayscale image by regression, classification, and adversarial learning, respectively.

3.1. Regression

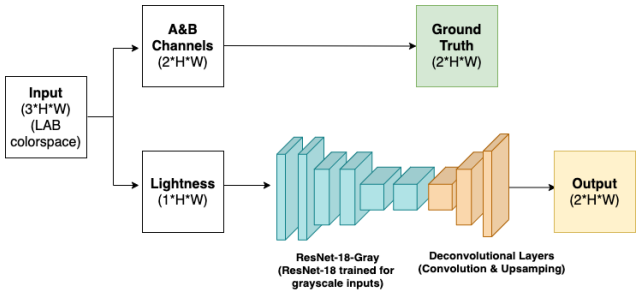

Figure 2. Regression Network Architecture

Inspired by semantic segmentation, we want to use an encoder-decoder style architecture to tackle this regression problem. We first use several convolutional layers to extract semantic information from the input images and then apply deconvolutional layers to upscale the extracted information. Specifically, the beginning of our model is a ResNet-18 ([5]), an image classification network with 18 layers and residual connections. We modify the first layer of the network to accept grayscale input images and cut it off after the 6th set of layers. It predicts a two-element vector (AB channel) for each pixel of the image at the end of the network, as demonstrated in the figure 2.

We want to minimize the Euclidean error between the AB channel we estimate and the ground truth. However, this loss function is problematic for colorization due to the multimodality of the problem since there may be multiple plausible solutions. As a result, our model will usually choose desaturated colors which are less likely to be "very wrong" than bright, vibrant colors. We optimize the loss function with the Adam optimizer. We train our model on 10k images with learning rate 1e-2, weight decay rate 1e-3, training batch size 256, and validation batch size 32 in 50 epochs. Batching the data at each epoch, then doing a forward pass to obtain the probabilities. We then calculate the loss and perform the backward pass, apply gradient descent and update the trainable parameters.

[2]https://tinyclouds.org/colorize/

3.2. Classification

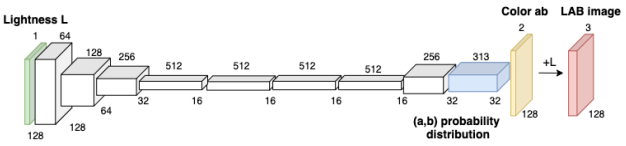

Figure 3. Classification Network Architecture

We improve our regression method into a classification to solve the multimodality problem. We quantize the AB space of the LAB color space into 313 bins and find a bin number between 0 and 312 for every pixel. The color prediction task is now a multinomial classification problem where every gray pixel can choose its AB channel from 313 classes. Inspired by the architecture proposed by Zhang et al. ([9]), we use a single-stream, VGG-styled network with added depth, dilated convolutions, and deconvolutional layers ([1],[8]). Each block has two or three convolutional layers followed by a Rectified Linear Unit and terminating in a Batch Normalization layer, as demonstrated in the figure 3.

We use a multinomial cross-entropy loss as our objective function. Let the output of the CNN be Z given an input image X. We transform all color images Y in the training set to their corresponding Z value. For every pixel in the original colored image $Y_{h,w}$, we find the nearest quantized ab bin and represent $Z_{h,w}$ as a one-hot vector. Since soft-encoding works well for training, we find the 5-nearest neighbors to $Y_{h,w}$ using KNN and weight them proportionally to their distance from the ground truth using a Gaussian kernel with $\sigma = 5$. Since colors' distribution in ImageNet is heavy around the gray line, we need to modify the standard cross-entropy loss into

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

where 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. We optimize the loss function with the Adam optimizer. Due to the limit of GPU memory, we train our model on 8k images with learning rate 1e-3, weight decay rate 1e-3 in 5 epochs. However, we did not successfully output the expected colored image after training the model. We summarize the potential reasons for the failure are (1) an inadequate number of training data (comparing to the work done by Zhang et al. [9], they trained the whole model with 1.3M pictures); (2) incorrect implementation in the objective function. We plan to tackle this problem in future work.

3.3. GAN

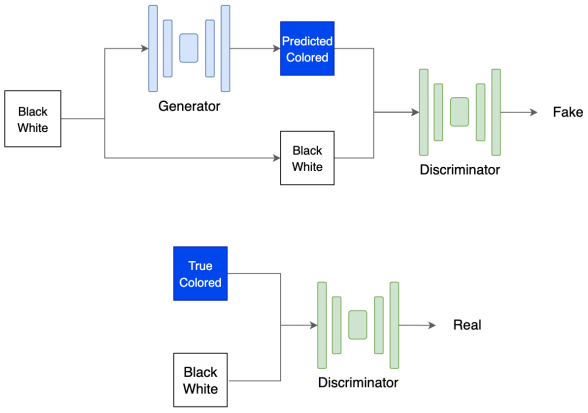

Figure 4. The pipeline of our GAN model

The third model we used is Generative Adversarial Network (GAN) ([3]). Unlike the previous two models, this model contains two neural networks: a generator and a discriminator. The generator works like previous models, takes in a black-white image and outputs the predicted colored image. The discriminator is trained to classify the predicted color image as false and the ground-truth color image as true.

The pipeline of the model on a black-white image and the corresponding colored image is shown in the figure 4.

For the generator, we adapt the similar model from ([6]), which is a ResNet-like ([5]) network. The generator is composed of an encoder and a decoder. The encoder has a sequence of layers like this,

$$C64-C128-C256-C512-C512-C512-C512-C512$$

The C{n} layer here means Convolution-BatchNorm-ReLU layer with n filters. All conv layers have filter 4 x 4 and stride 2, which downsample the image by 2. The Relu in the encoder is leaky with slope 0.2. The decoder has a sequence,

$$C512-C512-C512-C512-C256-C128-C64-X$$

First 3 layers also contain a dropout layer of 50%. Lastly, the X layer is an additional conv layer from 64 to 2 (output channel for AB).

For the discriminator, we also adapt the similar model from ([6]). It has a sequence of layers like this,

$$C64-C128-C256-C512$$

The Relu is leaky with slope 0.2. After the last conv layer, a convolution layer is added to produce a 1-dimensional output. Each value is then applied by a sigmoid function. The output would be a 1-dimensional image with value 0 or 1, indicating each pixel is fake or real.

In terms of loss function, there are two main loss functions applied here. First, the loss of the discriminator is computed as the average of its loss on a true image and its loss on a fake image. Each loss is a MSE loss for each pixel on the image (labeled 0 or 1). This loss could effectively capture whether this discriminator could distinguish between ground-truth image and generated image. Second, the loss of the generator is λ times L1 loss between predicted image and true image, plus discriminator loss on predicted image with label true. This loss could make the generator generate images that have less difference with true image and less loss from the discriminator.

During training periods, we would call forward(), backward() on D, and backward() on G. This serves a training procedure for each epoch. Then we perform training on our dataset, the hyper parameters are listed here.

Data size: 11849, Batch size: 32, Learning Rate: G: $2 \cdot 10^{-4}$, D: $2 \cdot 10^{-4}$, lambda on L1 loss: 500, Optimizer: Adam

4. Experiments

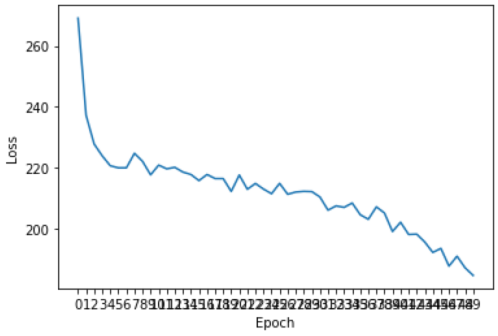

Figure 5. The training loss for regression

We randomly choose around 4k images in the ImageNet as our test dataset. We pre-process our training data to have higher colorfulness scores so the trained model can learn more about the semantic and textural information of

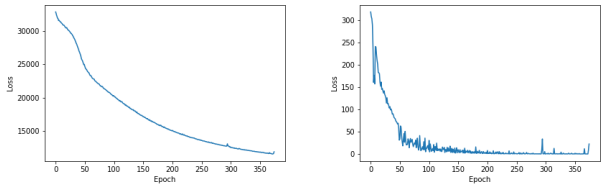

Figure 6. The training loss for GAN: Generator

the images and predict more reasonable colors on the input grayscale images. Due to randomness, if the test dataset has the images in the same categories as the training dataset, then the test results can tell us how well our model learn the information within a specific category; if the test dataset has the images not in the same categories as the training dataset, then the test results can tell us how well our model performs on the generalization.

We utilize three evaluation metrics in total and the summary of the results are shown in the table.

Firstly, structural similarity index measure (SSIM), a method quantifying differences in structure, contrast, and luminance. One advantage of SSIM is that it correlates to the human visual system (HVS) since human perception is very sensitive to structural change[7].

Secondly, we calculate the raw accuracy (AuC): the percentage of the predicted pixel is within a certain threshold compared to the ground truth pixel in ab color space. Specifically, we calculate the L2 distance between the predicted pixel and original pixel. Sweeping the threshold from 1 to 100, we have percentages of pixels within these thresholds, and the area under the curve (AuC) is calculated.

Lastly, we use human evaluations. For colorization problems, the ultimate goal is to colorize a grayscale image, making it plausible to a human observer. Therefore, to test each model's performance, it is essential to let humans manually evaluate the quality. In detail, 100 images are taken out from each model's test results and evaluated by human graders from a 1 to 5 scale. All images are randomly assigned to each grader to ensure fairness.

From table 1, we can see that regression has higher SSIM and AuC than GAN, but less human evaluation score than GAN. We think the reason is that regression is minimizing pure loss so the sum of L2 norm between predicted image and original image would be less, but the predicted may not be visually similar to the original image. However, the GAN has a discriminator instead of a pure loss so it could make the predicted image more visually reasonable. Overall, the GAN model carries out the colorization task better.

5. Implementation

We write the image preprocessing code for regression and classification models, which mainly focuses on converting the image in RGB format into LAB colorspace and separating the lightness channel from the other two channels. We also create a class object for creating data loaders. We adapt the code for reading FacadeDataset in the homework.

For the regression model, we adapt some of the network code from this repo [3]. For the classification model, we adapt some of the network code from this repo [4] and this repo[5]. We write all the training and test code on these two models.

For the GAN model, we adapt some of the GAN code from this repo [6]. The network is adapted from [6]. Besides, all the training code on GAN model is written by us.

	SSIM	AuC	Human
Classification	/	/	/
Regression	0.99896	0.97000	2.21
GAN	0.92393	0.72781	2.63

Table 1. Evaluation Metrics

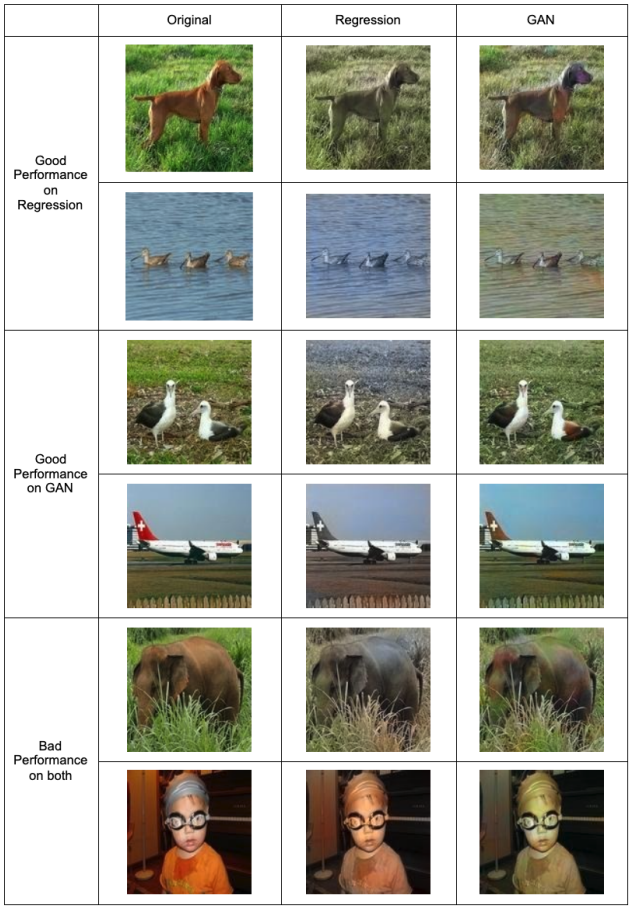

Figure 7. Results

[3]https://lukemelas.github.io/image-colorization.html
[4]https://github.com/richzhang/colorization
[5]https://github.com/hsalhab/Coloring-in-the-Deep
[6]https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

References

[1] Liang-Chieh Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:834–848, 2018. 2

[2] Zezhou Cheng, Q. Yang, and Bin Sheng. Deep colorization. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 415–423, 2015. 2

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. 2014. 1, 3

[4] David Hasler and Sabine E. Suesstrunk. Measuring colorfulness in natural images. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, *Human Vision and Electronic Imaging VIII*, volume 5007, pages 87 – 95. International Society for Optics and Photonics, SPIE, 2003. 1

[5] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2, 3

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. 2018. 1, 3, 4

[7] Peter Ndajah, Hisakazu Kikuchi, Masahiro Yukawa, Hidenori Watanabe, and Shogo Muramatsu. Ssim image quality metric for denoised images. pages 53–57, 11 2010. 4

[8] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2016. 2

[9] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. 2016. 1, 2, 3