Colorization Model Comparison

Churong Ji, Tianyi Li, Xiaoying Tang, Yixin Shi
University of Michigan
Ann Arbor, Michigan, MI 48109
{churong, litianyi, xytang, esing}@umich.edu

1. Introduction

Colorization on grayscale images is one of the important branches of image processing and computer vision with a wide range of applications. Photographs shot by Polariod cameras are only preserved in grayscale, but they will be more vivid and attractive if being colorized. Image colorization can also substitute for labor coloring, in fields such as manga coloring, image editing, and scene modeling, all of which are time-consuming and laborious. Our interest focuses on various techniques, mainly using deep learning, to solve the colorization problem with several metrics proposed for evaluation. We compared three different methods (GAN using RGB channels, CNN using RGB channels, CNN using LAB channels) and applied various metrics to evaluate the models' performances from different perspectives.

2. Related Works

Image colorization has experienced long-term progress with numerous significant leaps. Primarily, research solves the problem using traditional machine learning methods by matching luminance and texture information [4], or by scene recognition and segmentation [3], etc. These methods focus on traditional image processing techniques, while not implementing the state-of-the-art deep learning methods, consequently having low generalization capability. With the prosperity of the neural networks, researches gradually shift focuses to deep learning models, from basic fully connected layers [2], convolutional layers [6], and generative adversarial networks [5].

Enlightened by the works, we tried to develop three different models for image colorization, respectively a GAN model, a CNN model with RGB channels, a CNN model with LAB channels.

3. Method

In image colorization, our goal is to produce a colored image given a grayscale input image. This problem is challenging because it is multimodal — a single grayscale image may correspond to many plausible colored images. The deep neural network has shown remarkable success in automatic image colorization may in part be due to its ability to capture and use semantic information in colorization. We built three machine learning models (GAN, CNN Model with RGB channel, CNN Model with LAB channel) that automatically turn grayscale images into colored images.

3.1. GAN Model

First of all, we applied the Genarative Adversarial Network (GAN) containing two basic neural network models, respectively a generator and a discriminator. The generator is mainly trained to input a grayscale image and produce a predicted colorized one, while the discriminator is aimed at differentiating the ground truth from the generated image. More specifically, the generator is composed of an encoder and a decoder. The encoder of the generator aims at extracting the features from the image and concentrate the information by down-sampling. Visually, the encoder follows the pipeline (Figure 1)

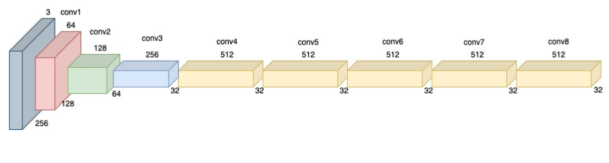


Figure 1: Encoder of the generative model.

where each C_N layer composes of a convolution layer, a batch norm layer, and a leaky ReLU activation layer. Specifically, the convolution layers consists of a 4 by 4 kernel, a stride of 2, and padding of 1. The leaky ReLU layer composes a negative slope of 0.2.

The decoder of the generator aims at regenerate the colorized image of the same size with the original grayscale image, and thus it will upsample the layer resulting from the encoder. Visually, it follows the pipeline (Figure 2)

Moreover, we connected the symmetric layers with same size in the encoder and decoder to avoid information loss due to downsampling in the encoder side. Intuitively, this

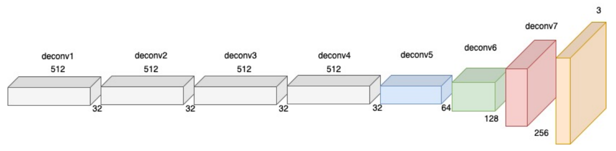


Figure 2: Decoder of the generative model.

will help shaping the rudimentary output with same textures and composition, while just lacking in color. A visualization of the model follows the pipeline (Figure 3)

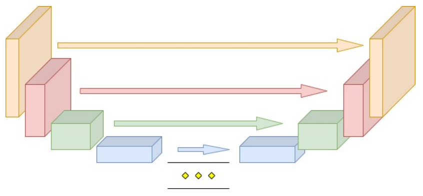


Figure 3: Combined Generative model.

The discriminator follows the pipeline (Figure 4)

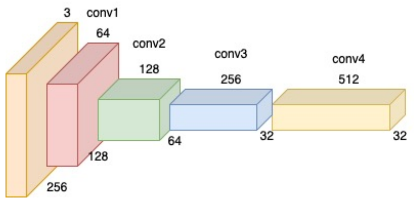


Figure 4: Decoder of the generative model.

The last convolution layer is followed by a sigmoid function to map the value into [0, 1]. All values less than 0.5 will be classified as fake image, otherwise as the ground-truth image.

To train the model, we defined the loss function as

$$L_{GAN} = L_{BCE} + \lambda L_1$$

where BCE loss is defined as

$$L_{BCE} = -\frac{1}{N}\sum y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

where y is the ground-truth pixel, and $\hat{y}$ is the predicted pixel. The L_1 loss is defined as

$$L_1 = \sum |y - \hat{y}|$$

Finally, the hyper-parameter $\lambda = 100$.

3.2. CNN Model with RGB Channels

Convolutional Neural Network (CNN) is a type of neural network model which allows us to extract higher representations for the image content. Though it is usually used for image classification, in this project, we are curious about how CNN works on colorization. Partially inspired by the model in [1], we implemented a similar 8-layer CNN model as shown in Figure 5. Here the input is the three channels of a 256×256 grayscale image (all three channels have the same value). Each conv layer in the figure represents a block of 2 or 3 repeated conv and ReLU layer, followed by a BatchNorm layer. We set the dilation of conv as 2 for conv5 and conv6 while use default dilation for other conv layers. All changes in resolution are achieved through spatial downsampling or upsampling between conv blocks. This model directly outputs the RGB channel of predicted images.

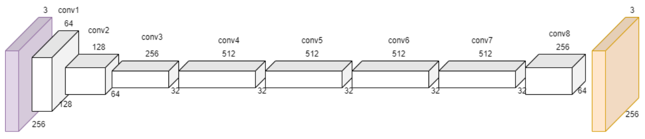


Figure 5: Architecture of CNN model with RGB Channels

We want to minimize the the L1 loss between the predicted RGB value and ground truth. Thus, we chose L1 loss as the loss function when training the model. We optimize the loss function with Adam optimizer. We trained our model on 500 images with learning rate 0.0002, exponential decay rate for the first moment 0.5 and exponential decay rate for the second moment 0.999. We set the training batch size 4 and trained the model for 30 epochs. In each epoch, we did a forward pass, calculated the loss and updated the weights of the model.

3.3. CNN Model with LAB Channels

LAB color space is another way to represent color with three channels: L channel represents the lightness of the color (L = 0 yields black and L = 100 indicates diffuse white); a represents its position between red and green (negative values indicate green and positive values indicate red); b represents its position between yellow and blue (negative values indicate blue and positive values indicate yellow).

In this model we aim to infer a full-colored image, which has 3 values per pixel from a grayscale image, which has only 1 value per pixel (lightness only). For simplicity, we only worked with images of size 128×128, so our inputs are of size $128 \times 128 \times 1$ (the lightness channel) and our outputs are of size $128 \times 128 \times 2$ (the other two channels).

Rather than work with images in the RGB format, we worked with them in the LAB colorspace (Lightness, A, and B) . This colorspace contains exactly the same information

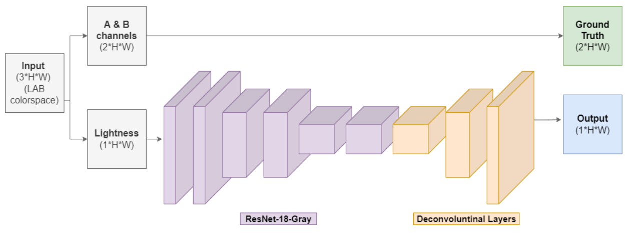


Figure 6: Architecture of CNN with LAB channels model

as RGB, but it will make it easier for us to separate out the lightness channel from the other two (which we call A and B). We made a helper function to do this conversion. We tried to predict the color values of the input image directly by regression.

Our model is a convolutional neural network[1]. We first apply a number of convolutional layers to extract features from our image, and then we apply deconvolutional layers to upscale (increase the spacial resolution) of our features.

Specifically, the beginning of our model is ResNet-18, an image classification network with 18 layers and residual connections. We modified the first layer of the network so that it accepted grayscale input rather than colored input, and we cut it off after the 6th set of layers.

Since we are doing regression, we use a mean squared error loss function: we minimize the squared distance between the color value we try to predict, and the true (ground-truth) color value. We optimize our loss function (criterion) with the Adam optimizer. We train our model with learning rate 1e-2, training batch size 4, and validation batch size 5 in 30 epochs.

As we want images in the LAB space, we first defined a custom data loader to convert the images. Next we define transforms for our training and validation data. After validation, we convert the generated images from LAB space back to RGB.

4. Experiments

4.1. Data Processing

For the dataset, we applied the standard benchmark dataset Natural Color Dataset (NCD) introduced in [1], containing 723 images of 20 categories. We chose it for model training and experiments because its size is small and it is easy to visualize and compare results. We split the data set to 580 training images, 143 testing images. Each image has size 256 $\times$ 512, where the first half is the grayscale image and the second half is the RGB image as shown in figure 7. One thing noteworthy is that images are resized to 128 $\times$ 256 for the CNN model using RGB channels.

[1]https://lukemelas.github.io/image-colorization.html

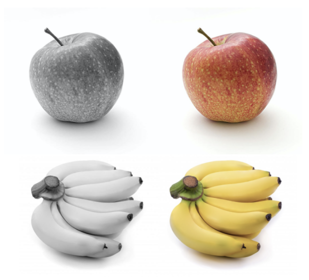
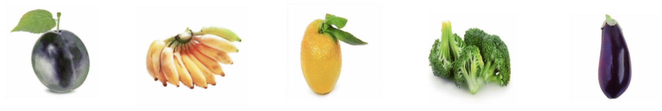


Figure 7: Natural-Color Dataset

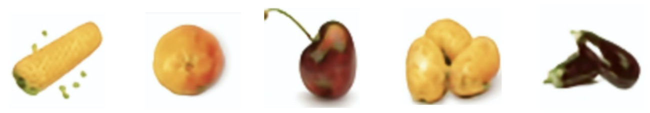
4.2. Evaluation

4.2.1 Sample Analysis

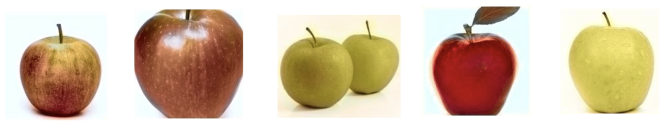
The whole dataset is used for either training or testing. The outputs are presented in figure 8.



(a) Colorized images by GAN with RGB



(b) Colorized images by CNN with RGB



(c) Colorized images by CNN with LAB

Figure 8: Colorization results

For the GAN model, although a majority of the validation data are performing well, some of them failed to generate the output with similar colorization. One of the illustrations is the left-most peach in figure 8a, which is colorized major in purple. This might result in the fact that peach is reasonably similar with plums in grayscale, and the GAN model misclassified the peach into a plum. On the other hand, since we are utilizing a combination of BCE loss and L_1 loss, instead of penalizing the RGB difference strictly, the model tolerates the deviation and outputs a plum-like colorization.

Our predicted result on CNN model with RGB channel has its outputs blurrier than inputs. We guess this is because the downsampling process overlooks some features of the image while the upsampling process fails to recover them.

The mean squared error loss function we implemented in the CNN model using LAB channels is slightly problematic for colorization due to the multi-modality of the problem. For example, if a gray fruit could be red or blue, and our model picks the wrong color, it will be harshly penalized. As a result, our model will usually choose desaturated colors that are less likely to be "very wrong" than bright, vibrant colors.

4.2.2 MSE and SSIM Scores

We compare the performances of the three models from three perspectives: training loss curve, mathematical metrics (Mean Squared Error and Structural Similarity Index), and human evaluation.

Firstly, as shown in figure 9, we observe the GAN model converges the fastest and comparatively smoothly while the CNN model using LAB channels converges but its training losses oscillate at the last epochs. The performance of the CNN model using RGB channels is between the other two models.

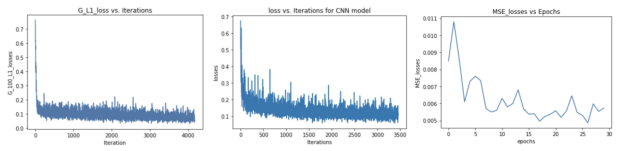


Figure 9: Training losses: GAN with RGB vs CNN with RGB vs CNN with LAB

Moreover, we select Mean Squared Error (MSE) and Structural Similarity Index (SSIM) to evaluate the quality of the colorized outputs. MSE gives straightforward evaluation of the colorization by comparing pix-to-pix values while SSIM can provide insights on structural information. Furthermore, the lower the MSE is, the closer the prediction image to the ground truth is, and the higher the SSIM is, the better the prediction image is in human perception according to this metric. As shown in table 1, the CNN model using LAB channels performs the best based on both metrics. However, the CNN model using RGB channels receives the lowest scores, for which we guess is because of its generated images' blurry edges as shown in figure 8.

	MSE	SSIM
GAN with RGB	3.16	0.81
CNN with RGB	6.39	0.68
CNN with LAB	1.53	0.85

Table 1: Model Performances based on MSE and SSIM

4.2.3 Human Evaluation

We included human evaluation to access the outputs. We saved the outputs of 50 same input images from the three models and randomly shuffled the order. Five human grader were invited to grade those works with a scale from 1 to 5, where 1 means obviously artificial and 5 indicates the image looks genuine. The results are presented in table 2, from which we see a contradiction of the grading based on mathematical metrics and human graders. Evaluated by the best model by MSE and SSIM, the CNN model using LAB channels is only scored 1.582 while the GAN model is rated as the best. From this we find the previous two mathematical metrics may not generalize well to our tasks. Moreover, we guess GAN receives the highest score because its mechanism tries to deceive the discriminator, just as trying to deceive human graders in this task.

	Score
GAN with RGB	3.613
CNN with RGB	2.448
CNN with LAB	1.582

Table 2: Model Performances according to human graders

5. Conclusion

In this work, we compared three different methods: GAN using RGB channels, CNN using RGB channels and CNN using LAB channels. Then, we applied various metrics to evaluate the models' performances from different perspectives. The CNN model using LAB channels receives the highest MSE and SSIM scores while the GAN model using RGB channels is evaluated the best by human graders, from which we observe that the mathematical metrics failed to generalize to our tasks. In the future, we plan to train our models on more complex datasets and research on how transfer learning works in those models. Moreover, we expect to devise metrics that align with human perceptions so that the contradiction between the scores by metrics and human graders can be addressed.

References

[1] Saeed Anwar, Muhammad Tahir, Chongyi Li, Ajmal Mian, Fahad Shahbaz Khan, and Abdul Wahab Muzaffar. Image colorization: A survey and dataset. 2020.

[2] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. *IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[3] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *Proceedings of International Conference on Computer Graphics and Interactive Techniques*, pages 689–694, 2004.

[4] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. *29th annual conference on Computer graphics and interactive techniques*, pages 277–280, 2002.

[5] Y.Ci, X.Ma, Z.Wang, H.Li, and Z.Luo. User-guided deep-anime line art colorization with conditional adversarial networks. *26th ACM international conference on Multimedia,*, pages 1536–1544, 2018.

[6] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *European conference on computer vision*, pages 649–666, 2016.