Generative Approach for Image Colorization

Zhongqian Duan* Chenyang Ma* Ruipu Li* Fengyu Yang* Qiming Ying*
University of Michigan
{duanzq, dannymcy, liruipu, fredyang, yingqm}@umich.edu

1. Introduction

Image Colorization is a challenging branch of computer vision that is extensively used in our daily life. With a given grayscale one-channel image, we will output a colored three-channel RGB image. Our goal is to produce a colorized image that is acceptable by human eyes. The output should be as realistic as possible but not necessarily identical to the original image. The problem is intuitively simple at first glance since we know semantically that the sky is blue and the cloud is white; however, for other objects including books, tables, computers etc., it is hard to estimate their color by their semantic meanings. In addition, factors such as changes in illumination, variations in viewpoints, and occlusions also increase the difficulty of this problem [1]. Despite the difficulties, image colorization has a wide range of real-life applications. With this technique, we are able to restore aged or degraded images, transform black-and-white videos to colored versions, and better restore historical sites.

With rapid development of deep learning in recent years, diversified models have been researched to solve this problem. Cheng et al. [2] first introduced CNN with a combination of joint bilateral filtering as the post-processing step. Zhang et al. [3] presented the colorful image colorization network to colorize grayscale images which successfully fooled humans on 32% of their trials. More recently, Generative Adversarial Networks (GAN) has been introduced to tackle this problem. GAN is able to learn a loss that tries to distinguish whether an image is real or not while learning a generative model to output 'fake' images to minimize this loss, which can be applied to our problem. GAN has been proved to work successfully in image to image translation problems including edges to photo, aerial to map, and labels to street scene, which can be treated as a good reference for our task [4]. In our paper, we designed a GAN network for image colorization tasks and compared it with a traditional CNN network.

2. Approach

2.1. General Pipeline

From chosen datasets, we partition our datasets into a training set, a validation, and a testing set. We first convert images in RGB colorspace in range 0 to 255 to images in L*a*b* color space of size $(128 \times 128 \times 3)$ in range -1 to 1. The L channel will serve as the input to the models $(128 \times 128 \times 1)$ as grayscale images. We then use CNN and GAN models to train the data using ADAM as the optimizer, and compare their performance quantitatively using four evaluation methods: L1 loss, L2 loss, PSNR and SSIM.

2.2. Detailed CNN Model

For CNN, we choose ResNet-18-Gray from [9], and it is a model based on ResNet-18. The general structure of this model can be concluded as the following Figure 1:

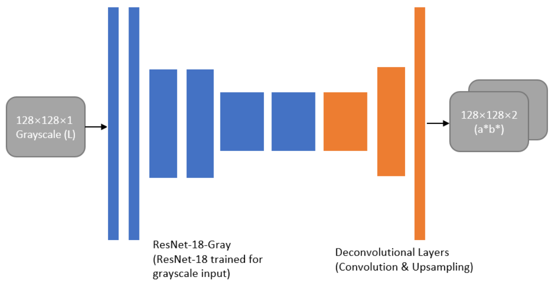


Figure 1: General structure of ResNet-18-Gray. Layers in blue are the convolutional layers, and layers in orange are the deconvolutional layers.

The input of this CNN model is grayscale images of size (128 × 128 × 1), the L channel, and the output is 2-channel a*b* colorized images of size (128 × 128 × 2). You can refer to ResNet-18 for detailed implementation of layers. From [9], mean squared error (L2) is chosen as the loss during training because it produces more vibrant colorization.

2.3. Detailed GAN Model

Our GAN model is a conditional GAN model with a generator and a discriminator. The difference between a GAN and a conditional GAN is that the generator of a GAN takes a random noise vector z as input, but conditional GAN takes z and grayscale images of size (128 × 128 × 1), the L channel, as inputs. The outputs is 2-channel images of size (128 × 128 × 2), the a*b* channels. The discriminator takes in these 2-channel images, concatenates them with the L channel, which is the generator input, and determines whether the generated colorized images successfully deceive itself as a real image. The output is a scalar.

The structure of the generator is illustrated in the following Figure 2:

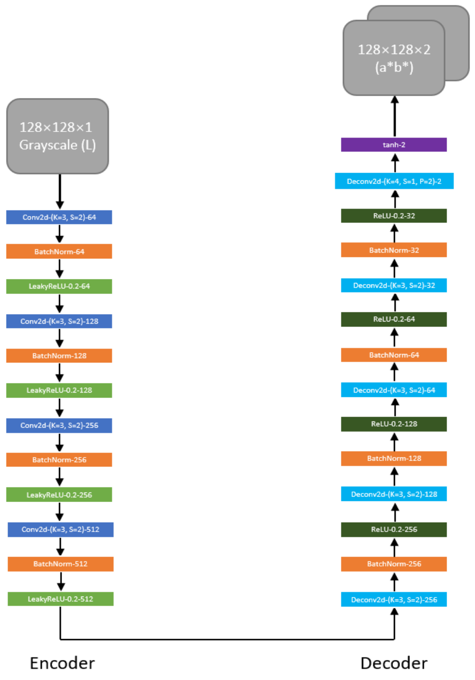


Figure 2: Detailed structure of the generator. Different colors represent different types of layers.

The structure of the discriminator is illustrated in the following Figure 3:

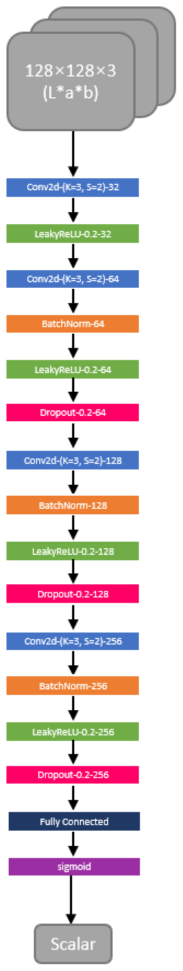


Figure 3: Detailed structure of the discriminator. Different colors represent different types of layers.

Both the generator and discriminator implement modules of the form Conv2d-BatchNorm-ReLu. The generator can be classified into an encoder and a decoder, and the decoder has tanh as the last activation function.

2.4. GAN Loss

The loss of this conditional GAN can be concluded as the following formula:

$$\mathcal{L}_{cGAN}(G,D) = E_{x,y}[logD(x,y)] + E_{x,z}[log(1 - D(x,G(x,z))] \quad (1)$$

Where G is the generator and D is the discriminator with their respective inputs and outputs with x, y, z to be the L channel, a* and b* channels, and the noise vector. In this model, z is not fed directly to G, but present in dropout layers. With the removal of the noise vector as input to G, some degrees of randomness is reduced; however, the addition of dropout layers, which randomly zeroes some of the elements of the input tensor, increases the randomness again. Although the addition of dropout layers is not necessary, it enables the generator to produce more creative and less deterministic outputs. Nevertheless, the balance between the randomness and determinacy of conditional GAN still requires much future work.

2.5. Loss of Generator

Given the experiences in [6], we add L1 loss to the generator loss with GAN loss. L1 loss will make our model more conservative because when the model attempts to reduce the L1 loss, it will tend to use gray-ish colors. L1 loss has the form:

$$\mathcal{L}_{L1}(G) = E_{x,y,z}[||y - G(x,z)||] \tag{2}$$

With the introduction of λ, GAN loss and L1 loss will be balanced, and our final objective, G*, can be described as:

$$G^* = arg \min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{3}$$

In which the purpose of the generator is to minimize the probability that the discriminator makes correct prediction on generated images, and the purpose of the discriminator is to maximize the probability of assigning correct labels.

2.6. Loss of Discriminator

From [6], one way to implement the loss of discriminator is by taking the average of the GAN loss of the real image and the GAN loss of the generated image, which can be expressed as:

$$0.5 \times (\mathcal{L}_{cGAN}Real + \mathcal{L}_{cGAN}Generated) \tag{4}$$

3. Experiments

We train our GAN network on the 3500 images randomly, validate on the 1000 images, and test on 500 images from the COCO dataset [5]. Even though the COCO dataset is designed mainly for object detection, we can make use of its abundant images taken from various circumstances in our daily life.

As mentioned in the introduction, it is hard to use any quantitative index to measure the performance of our model since our goal is to produce a colorized image that is acceptable by human eyes. Even Though the output may differ from the ground truth image, as long as it can fool human's eyes, it is a good output. However, it is hard to measure accurately how well the GAN model fools human's eyes. In order to compare with other models, we choose L1 loss, L2 loss, PSNR and SSIM four indexes quantitatively.

A visualization for selected images is shown in Figure 4. Quantitative measurements are shown in Table 1 below with comparison between GAN and CNN model

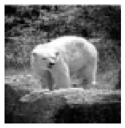
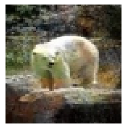
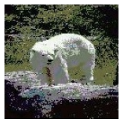
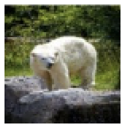
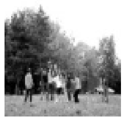
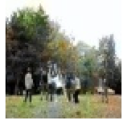
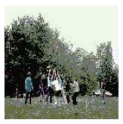
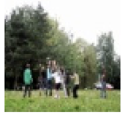
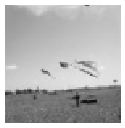
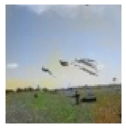
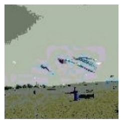
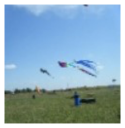
Input	**GAN**	**CNN**	**Ground Truth**
			
			
			

Figure 4: Example results from COCO dataset.

	L1	L2	PSNR	SSIM
GAN	4846186	31749	29.10	0.88
CNN	6880237	37858	28.31	0.71

Table 1: Test results of GAN and CNN model. Overall, GAN performs better than CNN. GAN has lower L1 and L2, which indicates that the generated images have lower errors compared to the real image. GAN has higher PSNR, which suggests the generated images have better quality and are more vibrant. GAN also has higher SSIM, which suggests there is a greater similarity between the generated images and the real images.

4. Implementation

For the implementation of the GAN network, we mainly code from scratch. The generative model was inspired by the encoder-decoder architecture from U-Net [8]. The implementation of “GAN loss” and “ensemble” and their mathematical theories were referenced from [6].

The CNN model serves as the comparison and reference to our GAN model. We implemented the CNN model based on [9] and its open source code.

5. References

[1]S. Anwar, M. Tahir, C. Li, A. Mian, F. Khan and A. Muzaffar, "Image Colorization: A Survey and Dataset", arXiv.org, 2017. https://arxiv.org/abs/2008.10774.

[2]Z. Cheng, Q. Yang and B. Sheng, "Deep Colorization", arXiv.org, 2015. https://arxiv.org/abs/1605.00075.

[3]R. Zhang, P. Isola and A. Efros, "Colorful Image Colorization", 2016. https://richzhang.github.io/colorization/resources/colorful_eccv2016.pdf.

[4]K. Nazeri, E. Ng and M. Ebrahimi, "Image Colorization Using Generative Adversarial Networks", 2017. https://arxiv.org/pdf/1803.05400.pdf.

[5]COCO - Common Objects in Context 2017, Cocodataset.org, 2017. https://cocodataset.org/#home.

[6]P. Isola, JY. Zhu, T. Zhou, "Image-to-Image Translation with Conditional Adversarial Networks", 2018. https://arxiv.org/abs/1611.07004.

[7]D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, "A. Efros, Context Encoders: Feature Learning by Inpainting", 2016. https://arxiv.org/abs/1604.07379.

[8]O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015. https://arxiv.org/abs/1505.04597.

[9] L. Kyriazi, G. Han, "Combining Deep Convolutional Neural Networks with Markov Random Fields for Image Colorization", 2016. https://arxiv.org/abs/1601.04589.