Image Colorization

Nick Grisanti

CS 153 Fall 2021

In a previous course I took at Harvey Mudd College, one of the options for a particular coding assignment was to develop a basic image colorization model. The idea was to take a grayscale image and return a prediction for what a color version of the image would look like. I opted for a different topic for this assignment and never dealt image colorization at all in the course, but the concept stuck with me as a cool idea. I figured I could use my project in CS 153 to give my best shot at developing an image colorization model and hopefully achieve results that would not have been possible in a standard homework assignment. It is with great pleasure that I finally present my image colorization model.

1 Original Work

1.1 My experience with the paper

This is an implementation project. I used the framework proposed in the research paper *Image-to-Image Translation with Conditional Adversarial Networks* by Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. As the title suggests, the paper describes a model for general "image-in, image-out" problems, such as changing the time of day of a photograph taken outdoors, filling in details of silhouettes, and of course image colorization. This does mean that the method I implemented was not designed *specifically* for image colorization, but this was intentional; after this project, I want to try transfer learning on the other tasks mentioned in the paper using the model I generated and see if the usefulness carries over.

The paper is quite comprehensive in its description of the network architecture and gave me a clear idea of the structure I needed to implement after my first read-through. I do have experience in machine learning so there was little terminology I had not seen before, but I did need to look up a few things for more detail. Specifically, I needed to do a bit more research on U-nets, which is one of the fundamental aspects of this paper's approach. The paper describes the general structure of a U-net and gave me enough information to know what to implement, but it used the terminology in a way that assumes the reader is familiar with it. I had never heard of a U-net before, so I was surprised to learn how common it is in medical image processing.

What the paper does not include, however, is information on the dedicated task of image colorization specifically. The tasks the authors test their model on all involve colorization to some capacity (e.g. filling in a sketch outline with details including color) and they show some colorized images from the dedicated image colorization experiment in one of the figures, but there is no other data on the task presented. Whereas loss information (i.e. choice of loss function, validation loss data, etc.) is presented for several tasks, this is not the case for dedicated image colorization, making it difficult to directly compare my results to those of the authors.

1.2 Summary

GANs The paper seems to assume the reader has a general understanding of GANs, but for the sake of clarity I will explain the how they work. A GAN, or a generative adversarial network, is an architecture for neural networks consisting of two smaller neural networks called a generator and a discriminator. The generator, as its name implies, generates the result the model is designed for, which in this case is colorized images. This generated output is often called a fake, since it was generated by a machine and is not real data. The fake, as well real data (which in this case are real color images) become the input of the discriminator, which outputs a probability that the item is fake. The loss function $\mathcal{L}_D$ for the discriminator on input image x can be written as

$$\mathcal{L}_D(x) = \begin{cases} \log(D(x)) & x \text{ is real} \\ \log(1 - D(x)) & x \text{ is fake} \end{cases}$$

where $D(x)$ is the output of the discriminator on input x. The generator is intended to fool the discriminator, so to speak, by producing fakes that discriminator will classify as real. The generator's loss function $\mathcal{L}_G$ is designed to reflect this:

$$\mathcal{L}_G(z) = \log(D(G(z)))$$

where z is the input to the generator, which is often just noise but in this case is a grayscale image. These two loss values are added together to find $\mathcal{L}_{GAN}$, which is the loss of the whole network. As a whole, a GAN can be described as breaking a large optimization problem into two smaller optimization problems in competition with each other. When training is complete, the scientist is left not only with a model that can generate realistic fakes for a particular task, but also with a model that can tell the reals and the fakes apart.

Conditional GANs The model described by the paper makes use of a variation of a GAN called a conditional GAN, or cGAN. This model gives the discriminator some extra information, namely the input z to the generator. Thus, the loss function for the discriminator becomes

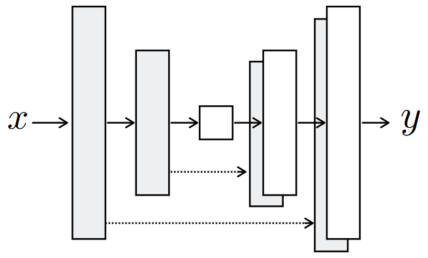

Figure 1: Architecture of a U-net

$$\mathcal{L}_D(x) = \begin{cases} \log(D(x,z)) & x \text{ is real} \\ \log(1 - D(x,z)) & x \text{ is fake} \end{cases}$$

where $D(x, z)$ is the output of the discriminator on inputs x and z. In this example, the discriminator is given two pieces of information—a grayscale image and a possible color version of it—and is tasked with determining if the color version is real or fake.

U-nets This covers the macro-level structure of the GAN, but the paper has specifications for the micro-level structure as well. Recall that the generator and discriminator are both neural networks themselves, and neural networks come in a variety of shapes. The model specified in the paper gives the generator a U-net architecture. A U-net consists of several downsampling layers followed by an equal number of upsampling layers, with skip connections between layers of the same size in the front and back halves of the network. The skip connections make low-level image information available to the deep layers of the network as they reconstruct the image. The upsampling half of the network contains several feature channels, which in this case correspond to the three color channels of the colorized image.

2 Implementation

2.1 Writing the code

I knew from the outset that this task would be rather computationally intensive, and my personal computer, despite having a decent GPU, would not be able to achieve the results I was hoping for in a reasonable amount of time. I decided to write my code in a Google Colab notebook to take advantage of Google's processing power. At first, I ran into issues of the runtime disconnecting mid-training, so I considered using XSEDE to run my code. After facing issues installing Fastai to my computer (which I have yet to resolve), I tried to find a way to get around the runtime issue on Colab. I was able to find a script to run in the console that tricks Colab into thinking I am active so it does not disconnect me from my runtime, and that ended up working.

A substantial portion of my code is adapted from that of Moein Shariatnia, who worked on a similar project. With my code base in place, I began adjusting the many hyperparameters of the model to achieve the desired result. I tried several loss functions, optimizers, network sizes, and even datasets until I was satisfied with the results; I ended up using L1 as the loss function, Adam as the optimizer, and Microsoft COCO as the dataset.

2.2 Results

Figure 2 shows the results of my implementation. Overall, the results appear quite accurate. The colorized kitchen image is nearly indistinguishable from the ground truth, for example. I am quite satisfied with these results and I am honestly surprised I was able to produce a model this accurate.

My model is not perfect, however, and serves as good evidence that perfect image colorization is likely impossible. Observe the blue tint of the "O" on the stop sign in the central image of Figure 2, and the incorrect hue of the man's face in the image to its right. A pattern I noticed in a substantial portion of the results is too sparing use of color, especially when the ground truth image contains objects of several different colors close to together in the frame. This implies the loss function penalizes a dull incorrect colorization less than a vibrant incorrect one, which intuitively makes sense; a dull picture of a bowl of fruit is more realistic than one with blue apples and red bananas.

Figure 3 shows the results of Isola et al.'s model. While the paper does not provide enough data for me to quantitatively compare my results to theirs, our results appear qualitatively similar. I must say though that I think my model is a little more accurate. Although my model is sometimes too conservative with its use of color, I never experienced it returning a result that looks completely grayscale like Isola et al.'s model.

Figure 2: Results of my image colorization model. The top row contains grayscale images, the middle row contains my model's colorization, and the bottom row contains the ground truth.

Figure 3: Result's of Isola et al.'s image colorization model. The top row contains grayscale images, the middle row contains my model's colorization, and the bottom row contains the ground truth.

References

[1] Isola, Phillip, et al., *Image-to-Image Translation with Conditional Adversarial Networks*, Cornell University, 26 November 2018.

[2] Shariatnia, Moein, "Colorizing black & white images", Towards Data Science, 18 November 2020.