CS 520: Assignment 4 - Colorization

16:198:520

The purpose of this assignment is to demonstrate and explore some basic techniques in supervised learning and computer vision.

The Problem: Consider the problem of converting a picture to black and white.

Figure 1: Training Data - A color image and its corresponding greyscale image.

Typically, a color image is represented by a matrix of 3-component vectors, where Image$[x][y] = (r, g, b)$ indicates that the pixel at position (x, y) has color (r, g, b) where r represents the level of red, g of green, and b blue respectively, as values between 0 and 255. A classical color to gray conversion formula is given by

$$\text{Gray}(r, g, b) = 0.21r + 0.72g + 0.07b, \tag{1}$$

where the resulting value Gray(r, g, b) is between 0 and 255, representing the corresponding shade of gray (from totally black to completely white).

Note that converting from color to grayscale is (with some exceptions) *losing* information. For most shades of gray, there will be many (r, g, b) values that correspond to that same shade.

However, by training a model on similar images, we can make contextually-informed guesses at what the shades of grey ought to correspond to. In an extreme case, if a program recognized a black and white image as containing a tiger (and had experience with the coloring of tigers), that would give a lot of information about how to color it realistically.

Figure 2: Trained on the Color/Grayscale image in Fig.1, recovers some green of the trees, and distinguishing blues between sea and sky. But there are definitely some obvious mistakes as well.

You have a lot of freedom in your approach to this, but carefully formulate each of the following in outlining your solution to the problem, expressing your design choices, the math, and the algorithms behind your solution:

- **Representing the Process:** How can you represent the coloring process in a way that a computer can handle? What spaces are you mapping between? What maps do you want to consider? Note that mapping from a single grayscale value **gray** to a corresponding color (r, g, b) on a pixel by pixel basis, you do not have enough information in a single gray value to reconstruct the correct color (usually).
- **Data:** Where are you getting your data from to train/build your model? What kind of pre-processing might you consider doing?
- **Evaluating the Model:** Given a model for moving from grayscale images to color images (whatever spaces you are mapping between), how can you evaluate how good your model is? How can you assess the error of your model (hopefully in a way that can be learned from)? Note there are at least two things to consider when thinking about the error in this situation: numerical/quantified error (in terms of deviation between predicted and actual) and *perceptual* error (how good do humans find the result of your program).
- **Training the Model:** Representing the problem is one thing, but can you train your model in a computationally tractable manner? What algorithms did you draw on? How did you determine convergence? How did you avoid overfitting?
- **Assessing the Final Project:** How good is your final program, and how can you determine that? How did you validate it? What is your program good at, and what could use improvement? Do your program's mistakes 'make sense'? What happens if you try to color images unlike anything the program was trained on? What kind of models and approaches, potential improvements and fixes, might you consider if you had more time and resources?

Some Possible Approaches

Some possible approaches you might take to the problem include the following (and where used to generate the small example above):

- While mapping from itgray $\mapsto (r, g, b)$ cannot reliably reconstruct the true color of a pixel, not having enough information in a single gray value, consider looking at a group or patch of gray pixels, and mapping this set of *set* of gray values to a single (r, g, b) color vector, which could for instance be the color of the middle pixel in this set. In this case, the surrounding gray values give additional context and information to build a color for the central pixel. With such a map, a grayscale image could be colored by simply taking every relevant pixel patch, and determining what color the central pixel should be. Could this be improved?
- To further simplify things, the problem can be shifted from a *regression* problem to a *discrete classification* problem in the following way: consider building an initial palette of K representative colors, and instead of trying to reconstruct the true color of a pixel, determine which of these K colors should best be applied to a given pixel. How can you determine which K colors are best to use, however? And be careful as well - how should you assess error and the quality of a model when coloring in this way?

Bonus

Imagine trying to reconstruct damaged images after an event like a fire. How could you build a system to 'in-paint' or fill in holes/missing areas in images? Consider the five areas outlined above in designing such a thing. Build it.