Colorization

CS 440

Cesar Herrera, Jae Weon Kim

Spring 2020

List of Figures

The problem

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

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

$$Gray(r,g,b) = 0.21r + 0.72g + 0.07b,$$

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 come exceptions) *losing* information. For most shades of gray, there will be many (r,g,b) values that correspong 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.

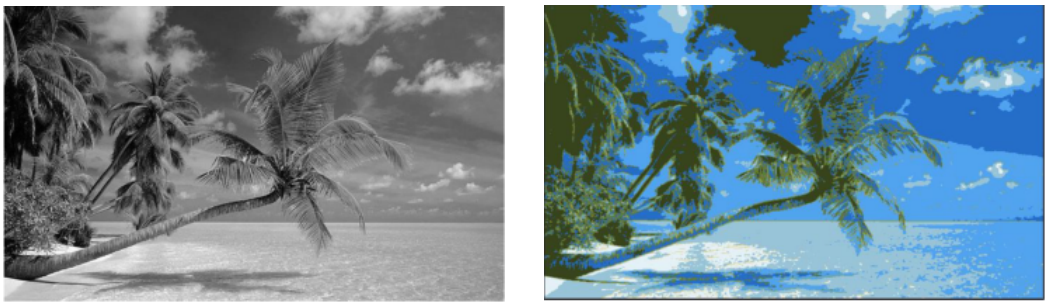

Figure 2: Trained on the Color/Grayscale image in Fig.1, recovers some green of the trees, and distinguishing blues between sea and sky. But there are definitely some obvious mistakes as well.

For the purpose of this assignment, you are to take a single color image (of reasonable size and interest). By converting this image to black and white, you have useful data capturing the correspondence between color images and black and white images. We will use the left half of the image as training data, and the right half of the image as testing data. You will implement the basic model described below to try to re-color the right half of the black and white image based on the color/grayscale correspondence of the left half and as usual, try to do something better.

The Basic Coloring Agent

Consider the following basic strategy for coloring an image: while a single gray value does not have enough information to reconstruct the original cluster, considering the surrounding gray pixels might. So given a 3x3 patch of grayscale pixels, we might try to use these 9 values to reconstruct the original (r,g,b) value of the middle pixel. We simplify this further in the following way:

- Instead of considering the full range of possible colors, run k-means clustering on the colors present in your training data to determine the best 5 representative colors. We will color the test data in terms of these 5 colors.
- Re-color the right half of the image by replacing each pixel's true color with the representative color it was clustered with.
- For every 3x3 grayscale pixel patch in the test data (right half of the image), find the six most similar 3x3 grayscale pixel patches in the

training data (left half of the image).

- For each of the selected patches in the training data, take the representative color of the middle pixel.
- If there is a majority representative color, take that representative color to be the color of the middle pixel in the test data patch.
- If there is no majority representative color or there is a tie, break ties based on the selected training data patch that is most similar to the test data patch.
- In this way, select a color for the middle pixel of each 3x3 grayscale patch in the test data, and in doing so generate a coloring of the right half of the image.
- The final output should be the original image, the left half done in terms of most similar representative colors to the original image colors, and the right half done in representative colors selected by the above process.

How good is the final result? How could you measure the quality of the final result? Is it numerically satisfying, if not visually?

Bonus: Instead of doing a 5-nearest neighbor based on color selection, what is the best number of representative colors to pick for your data? How did you arrive at that number? Justify yourself and be thorough.

The Improved Agent

In the usual way, we want to build an improved agent that beats the basic agent outlined previously. You have a lot of freedom in how to construct your approach, but follow these guidelines:

- Use the left half of the image as training data (color/grayscale correspondence) and the right half of the image as testing data (having converted it to grayscale).
- The final output should be the original image, with the right half colored according to your model.

- *The use of pre-built ML libraries or objects, or automatic trainers, is strictly forbidden. You can use TensorFlow for instance as an environment to build your model in, as long as you do not make use of layer / model objects or automatic differentiation /training*
- A specification of your solution, including describing your input space, output space, model space, error or loss function, and learning algorithm.
- How did you choose the parameters (structure, weights, any decisions that needed to be made) for your model?
- Any pre-processing of the input data or output data that you used.
- How did you handle training your model? How did you avoid overfitting?
- An evaluation of the quality of your model compared to the basic agent. How can you quantify and qualify the differences between their performance? How can you make sure that the comparison is 'fair'?
- How might you improve your model with sufficient time, energy, and resources?

As usual, be thorough, clear, and precise, with the idea that the grader should be able to understand your process from your writeup and data.

Some Possible Ideas

- The basic agent described previously executed k-NN classification, using the pre-clustered colors. You could also think of this as a regression problem, trying to predict the red/green/blue values of the middle pixel.
- Note that there are things you know in advance about the red/green/blue values, for instance, they are limited to values between 0 and 255. How does this inform your model?
- How could soft classification like logistic regression be used?
- Neural networks are always an option, but how should you choose the architecture?

- The k-NN classification of the basic agent essentially reduces the output space dramatically - how could the input space be similarly reduced?