# Colorizing Project

Frank Fondell

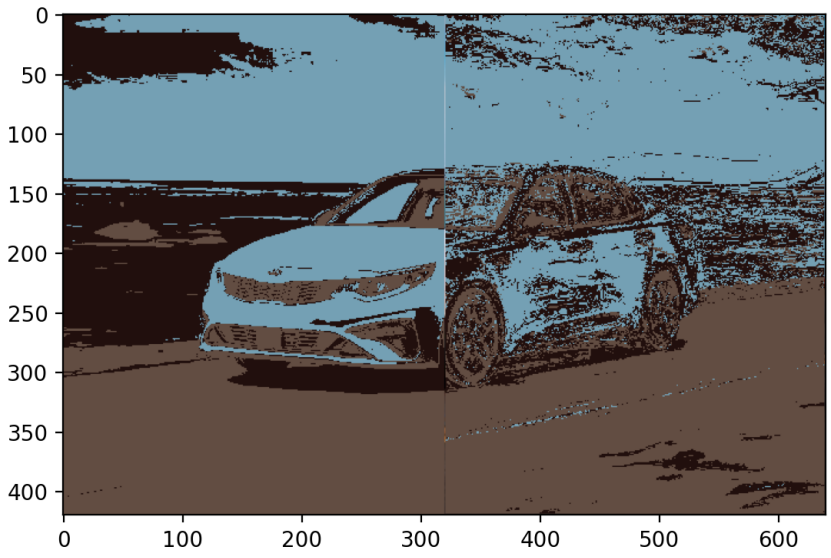
May 5, 2021

## The Basic Agent

When building the basic agent, by far the biggest challenge was having the algorithm run in a reasonable amount of time. This slowed down my progress immensely and made it difficult to make progress on the portion. When I was finished, the output was objectively terrible compared to the original, the right side that was generated vaguely resembled the picture on the left. I'd say the output is decently visually satisfying. To numerically compare the success of the right side image, I created a right side with representative colors and compared the error/loss to my generated image using the formula below.

$Error/Loss = \sum_{N=1}^{N}(f(x_i) - y_i)^2$

The error/loss is the sum of the euclidian distance squared between the RGB values of these two images, the closer it is to 0, the better the algorithm. This value came out to be 5622238.99 and the performance is displayed below. It seemed that images with a higher quality and complexity would result in a lower average distance.

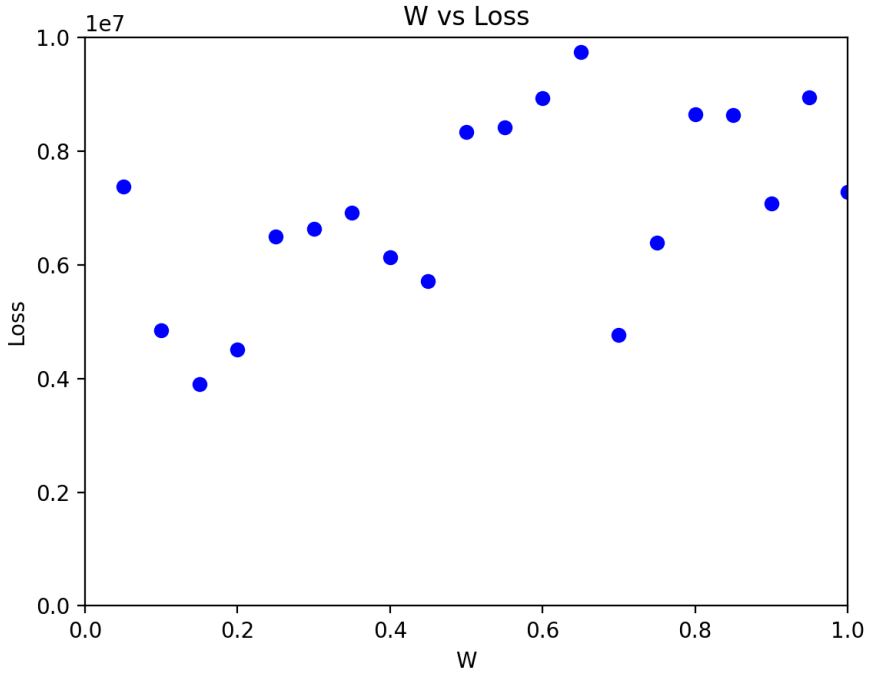


## The Improved Agent

Modeling the improved agent upon a model that is not linear is a good idea because it provides a model for a best fit that doesn't generalize about the data too much. Since we are comparing two images that are different, but contain the same type of objects, there should be a certain level of generalization but at the same time it would be detrimental to use a linear model that generalizes too much as these are two completely different images. In the improved agent, I modeled my calculation for nearest color value by using the below equation for each value, red, blue, and green.

$$RGB_i = B + W_1 * G_1 + W_2 * G_2 + ... + W_N * G_N$$

Compared to my basic agent, my improved algorithm was able to run much faster, and therefore opened up the ability to run many tests to calculate optimal values. Unfortunately, I didn't use my time wisely and was unable to implement any algorithms that actually allow for learning off the previous iteration. I did however do a test to find an optimal weighting factor for the current algorithm. On average, a lower weighting factor for all colors tends to produce a smaller loss as shown below.




Given more time, I would improve this algorithm using a new equation that changes the weights and bias based off the previous performance. This equation

would calculate the difference between the current and previous loss and use this margin to either encourage an increase or decrease in the weight factors by proportion to the difference in loss between iterations.