# Colorization Report

May 5, 2021

## Group Information

Reagan McFarland (rpm141)

Alay Shah (acs286)

Toshanraju Vysyaraju (tv135)

### Work Split

- Basic Agent: Reagan, Alay, and Toshanraju
- Improved Agent: Reagan, Alay, and Toshanraju
- Questions and Write Up: Reagan, Alay, and Toshanraju

### Honor Statements

- Reagan McFarland: On my honor, I have neither received nor given any unauthorized assistance on this assignment.
- Alay Shah: On my honor, I have neither received nor given any unauthorized assistance on this assignment.
- Toshanraju Vysyaraju: On my honor, I have neither received nor given any unauthorized assistance on this assignment.

## Image

Below is the image we decided to use:

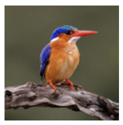


Figure 1: Original Image (128x128)

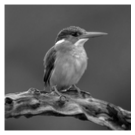


Figure 2: Grey Scale (128x128)

## Basic Agent

Output of 5 nearest neighbors algorithm:

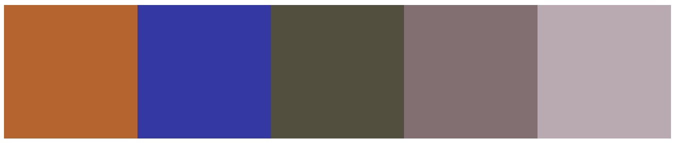


Figure 3: Five Colors used

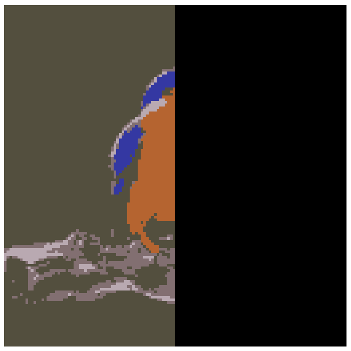


Figure 4: Output of 5 means (128x128)

**Final output using algorithm described in assignment:**

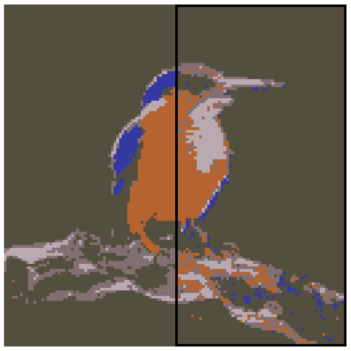


Figure 5: Output of the basic agent (128x128)

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

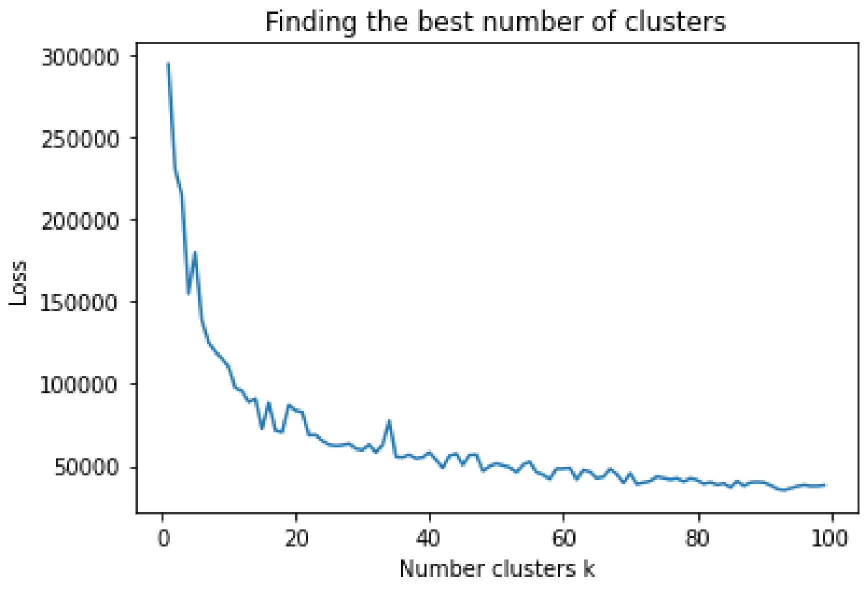
The final result is fairly good, with respect to only the five colors. We can make out the piece of wood the bird is on as well as the texture on it. We also can wee the bird's beak as well. However, it obviously lacks the details of the original picture since it has been reduced down to just 5 colors.

**Bonus: Instead of doing a 5-nearest neighbor based 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 best number of representative colors seems to be between 10 and 12 colors for our image. We arrived at this conclusion by computing the loss at each k and plotting it. The loss is computed as the sum of the distance of the RGB values to the color assigned using k means over the training half of the image.

$\text{Loss}_k = \sum_{i,j}^{n,n/2} \text{dist}(RGB_{i,j}, KColor_{i,j})$

In class, we discussed how the 'elbow', or where the function begins to decrease at a decreasing rate, will reduce the error as well as stop overfitting from occurring.



Below is the output for the k-nearest neighbors, when k = 11:

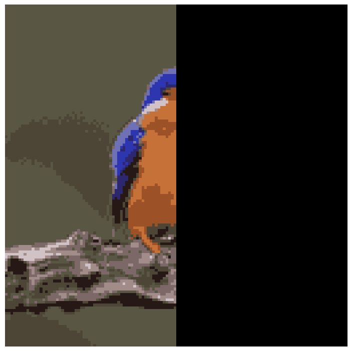


Figure 6: k nearest neighbors with k = 11

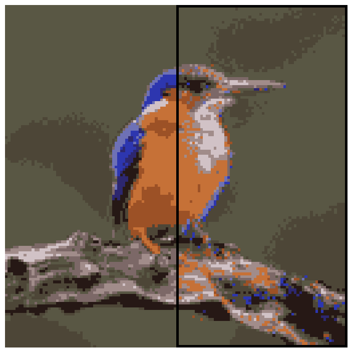


Figure 7: Basic Agent with 11 colors

## Improved Agent

**Disclaimer:** We, unfortunately, were unable to complete the Improved Agent due to a lack of time. However, below are our ideas and preliminary results:

### Our model:

We decided to use a non-linear regression model for each of the RGB values. This model makes use of the sigmoid function to map all input to a value between 0 and 1. The input of the model, $\vec{x}$, will be the 3x3 greyscale patch converted to a value between 0 and 1. The input space are values between[0,1]. The first term in $\vec{x}$ will be 1 to account for $\vec{w}_0$(bias). Then, features are added to make this linear input quadratic by $\vec{x}^T * \vec{x}$. Thus our input is a vector of length 100. The weights are set at random between (-0.5, 0.5) and are trained using the function described below. The weights are chosen between these values to ensure that the weights do not grow extremely large and start off near 0. The output for this function is one value in the space [0,1] for a specific color value for a specific pixel. Thus, we are able to take in 100 values in the space [0,1] and output one value in the space of [0,1]. There are three models, one for red, green, and blue.

#### Bonus: Why is using a linear model a bad idea?

We chose this model because just a linear model would output values outside the bound of [0,255]. With the sigmoid function, we are able to take in any value and have it bound between [0,1]. With post processing the data we are able to convert it to a value between [0,255] and therefore the model is much more useable.

#### General Idea

$R = \sigma(\vec{w}_R \cdot \vec{x}_i)$

$G = \sigma(\vec{w}_G \cdot \vec{x}_i)$

$B = \sigma(\vec{w}_B \cdot \vec{x}_i)$

Each model will be trained separately.

#### Data Preprocessing and Post Processing

We took the original RGB and the Greyscale matrices and normalized them by dividing by 255. When reconverting the image, we will multiply the output matrix by 255 to get the RGB values in a range [0,255] from the original output space of [0,1].

---

#### Model:

$F(\vec{x}_i) = \sigma(\vec{w} \cdot \vec{x})$

For red, our model would look like: $R = \sigma(\vec{w}_R \cdot \vec{x}_i)$

For green, our model would look like: $G = \sigma(\vec{w}_G \cdot \vec{x}_i)$

For blue, our model would look like: $B = \sigma(\vec{w}_B \cdot \vec{x}_i)$

#### Loss Function:

$L = \sum_{i=0}^{N} (F(\vec{x}_i) - y_i)^2$

Since we will be using Stochastic Gradient Descent, we are more interested in $L_i$

$L_i = (F(\vec{x}_i) - y_i)^2$

**Learning Algorithm (updating the weights):**

Then to update, we would want to find the value that minimizes this loss, i.e. find the derivative of $L_i$ with respect to $\vec{w}_t$.

$\frac{\partial L_i}{\partial \vec{w}_t} = 2(F(\vec{x}_i) - y_i) * \frac{\partial}{\partial \vec{w}_t}(F(\vec{x}_i) - y_i)$

$= 2(F(\vec{x}_i) - y_i)\frac{\partial}{\partial \vec{w}_t}(F(\vec{x}_i))$

$= 2(F(\vec{x}_i) - y_i)\frac{\partial}{\partial \vec{w}_t}(\sigma(\vec{w}_t \cdot \vec{x}_i))$

$= [2(F(\vec{x}_i) - y_i) * \sigma'(\vec{w}_t \cdot \vec{x}_i)]\vec{x}_i$

$= [2(F(\vec{x}_i) - y_i) * \sigma(\vec{w}_t \cdot \vec{x}_i) * (1 - \sigma(\vec{w}_t \cdot \vec{x}_i))]\vec{x}_i$ (We know the value of $\sigma'(z) = \sigma(z)(1 - \sigma(z))$)

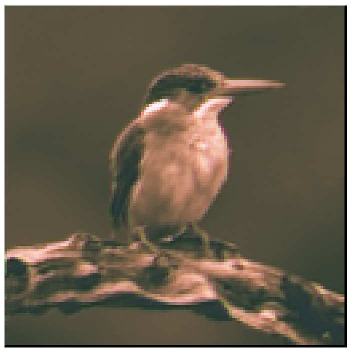
From this, we can get the formula to update the weight vector at time t+1 as:

$\vec{w}_{t+1} = \vec{w}_t - \alpha * [2(F(\vec{x}_i) - y_i) * \sigma(\vec{w}_t \cdot \vec{x}_i) * (1 - \sigma(\vec{w}_t \cdot \vec{x}_i))]\vec{x}_i$

## An Evaluation of our Model

Unfortunately, due to time constraints, we were not able to get a sufficiently good output from our model. We believe that it requires more experimentation and that we were not able to find a proper value for alpha. One possible route we could take is adding more features, we believe that passing the quadratic model into the sigmoid may not have been enough and we may need to revisit this. Our learning rates were chosen by graphing the values of the loss at each step, this ensured that we were able to see the loss converge. Unfortunately, the loss converged at too high of a value as seen in our output. The basic agent performed better than this parametric agent because it did not require as much fine tuning. Furthermore, the basic agent was only operating with five choices, where as the improved agent worked in a continuous 3-D space between [0,1]. With sufficient time and resources, we would improve by experimenting with different learning rates and adding more features until the output was both visually and numerically pleasing.

## Output of improved Agent:



# Bonus: Clever Acronym

Our acronym for the advanced agent algorithm is Advanced Colorization Algorithm for Birds (ACAB).

# Bonus: LaTeXed Report

This report was written and compiled using LaTeX.