CS: 440 Project 4 - Colorization

Amy Wang, Christopher Yeh

Basic Coloring Agent

Our basic agent colorizes the 150x150 photo by following the steps given in the assignment. Given an image, it is first converted to Grayscale using the equation Gray(r, g, b) = 0.21r+ 0.72g+ 0.07b. While going through each pixel, the (r, g, b) values for the left side of the image is recorded into a matrix. Then, k-means clustering is done on the left half of the picture to find the best 5 representative colors.1

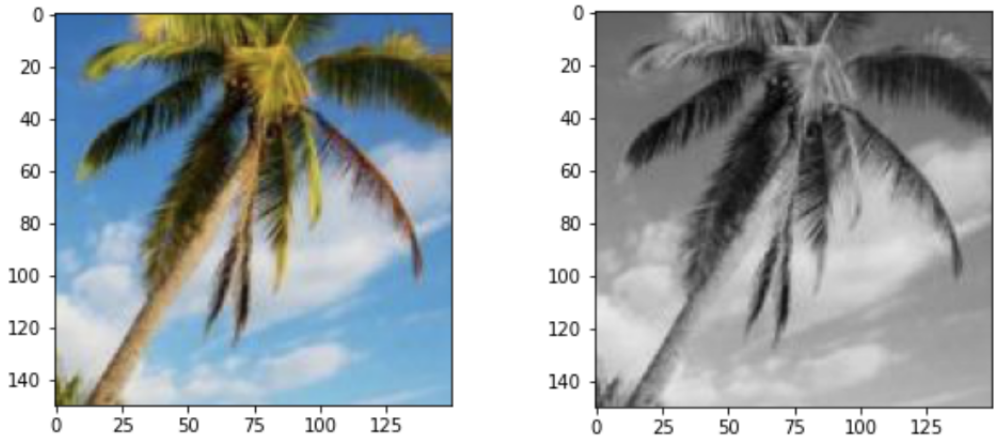

Figure 1: (left) Original Image, (right) Grayscaled Image

K-Means Clustering: In the beginning, 5 random (r, g, b) values are chosen to be the first 5 centroids. In each iteration, each pixel is first assigned to the closest centroid. Then, in the collection of values for each centroid, the values are averaged and a new centroid is chosen. The process repeats until the centroid values no longer change. The returned 5 (r, g, b) values are taken as the best 5 representative colors.

The pixels on the left side of the original image were then replaced with the nearest representative color from the clustering.

K-Nearest Neighbors: Following the instructions in the assignment, we convert every 3x3 grayscale pixel patch into a 1x9 vector. For each patch in the testing data (right side), we go through every patch in the training data (left side) and find the 6 patches where the distance between the two vectors is the smallest. The center of the patch is compared with the output in the training data and the corresponding color for said vector is found. If there is a majority representative color, the corresponding middle pixel in the test patch is changed to that color. If there is a tie, then the color corresponding to the training vector closest to the testing vector is what it will be changed to.

The final image is displayed with the left side changed to the 5 best representative colors and the right side changed to the predictions from the KNN model. A 1-pixel black band is applied to cover up the spots where we were not able to get 3x3 vectors of.

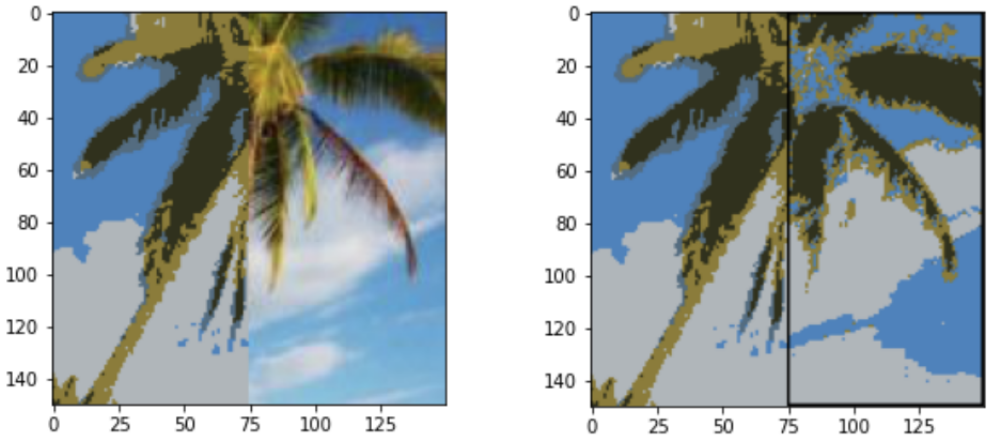

Figure 2: (left) Best Representative Colors, (right) Final Result

Result Evaluation: Visually speaking, the final result looks pretty good. There is clear distinction between the different objects with the exception of the blue part in the middle of the tree. However looking at the grayscale image for that part, the color is a very light gray - corresponding to a yellowish green in the original image despite it being a part of the tree. The 5 best representative colors do not include that color and so the closest shade of gray would be to the color of the sky. The mean square errors of R, G, B are calculated and the values are very high at around R: 1789, G: 2830, B: 1005. Although the result looks alright, mathematically it is not a very good model.

Bonus - Best Number of Representative Colors: In order to find the best number of representative values, we did k-means clustering with different values of k. The replaced image of K = 8 and K = 10 is displayed in the code. Ideally, we would run the basic agent on the image with the different values of k and see which one would produce the smallest error, but because our KNN ran rather slowly, we eyeballed the colors to make a decision. For our image, a K value of 8 would work well because it solves the issue from before where the small yellowish green area was not considered a representative color. Thus, if the basic agent were to predict off of the 8 colors, we would achieve a better result. We also plotted the 3D graph of the rgb values but because there are too many points, no conclusion can be made for how many clusters there are.

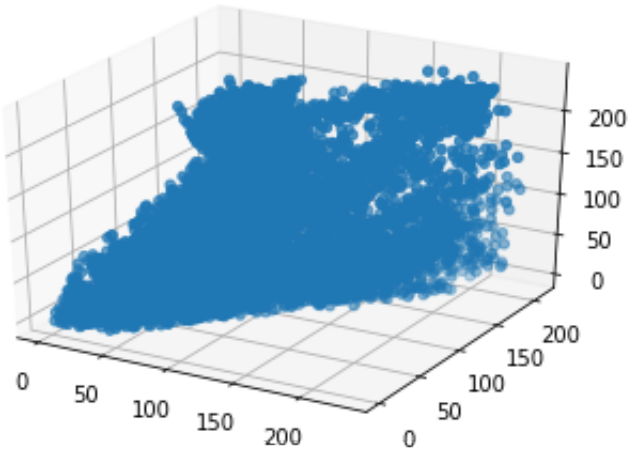

Figure 3: 3D graph of R, G, B Values

Improved ML under Logistic Regression

Since there is no any constraint of choosing certain number of colors through the grayscale dataset input (in which there is only one feature for the input data...), the utilization of regression for Machine Learning will be a perfect fit whereas a numerical input (grayscale spans from 0 to 255) also leads toward "several" numerical outputs (RGB spans from 0 to 255 for each attribute). Keep in mind that since there are 3 different outputs that are being observed/trained/tested throughout the Machine Learning Process, there will also be 3 mathematical models being generated among the training in which the correlation of Red/Green/Blue to Grayscale is interpreted individually.

Since the original linear regression (such as OLS) may not be able to generate a satisfying "line" in which such ordinary algorithm only establishes a "straight" line for the predicted regression. When it comes to photo analysis, it is the fact that a single numerical grascale does not give too much information (especially to this specific case when a single feature input is trying to be mapped to a 3 features output...). For instance, there may be extreme cases such as a low "Grayscale" value may include more information regarding "Blue" due to the low coefficient of the RGB to Grayscale formula.

With this in mind, for the improved ML model, the Logistic Regression model is adopted in which there are various parameters that can be manipulated on, therefore leading to a reasonable output by seeking for a best fit of the "curve" toward the training data points. Although the use of ML library is prohibited, there may still exist more efficient approach toward determining the mathematical model with the fact that all features and samples are spanned within [0:255].

Logistic Regression: Mathematical Model

By establishing a suitable mathematical model for Machine Learning toward this specific task, the general Sigmoid function for Logistic Regression can be initialized as follows:

$$\mathbb{P} = \frac{e^{\beta_o + \beta X}}{1 + e^{\beta_o + \beta X}}$$

whereas β_o represents the model bias, β determines the "curviness" of the curve, and X represents the input (in this case its the the numerical value of grayscale spanning between 0:255).

In order to generate a more precise/efficient model, the above formula may be reduced as follows:

$$\begin{aligned}\mathbb{P} &= \frac{1}{e^{-(\beta_o + \beta X)} + 1} \\ &= \frac{1}{1 + e^{-\beta(X+A)}}\end{aligned}$$

The modified formula above allows more convenient model establishment in which by adjusting A, it pretty much moves the curve horizontally.

With such mind-blowing moment & the fact that both input & output are constrain within [0:255], the graph can simple be moved horizontally by setting $255/2 = 127.5$ as the center of the curve. Additionally, since the original model converges at 1 when X gets toward infinity, such model may also be scaled up to 255 in which the predicted output is also meant to span between 0 to 255. Thus, the modified mathematical model can be rewritten as follow:

$$Y = \frac{255}{1 + e^{-\beta(X - 127.5)}}$$

Keep in mind that there are three Y outputs (R/G/B) that are yet to be determined, in which it leads toward training of 3 learning models with different β that corresponds to different color predictions.

Logistic Regression: Training ML Model

According to the mathematical model that has been obtained, the ML model is only down to determine one parameter: β. As mentioned, by adjusting such value, it changes the "curviness" of the line. However, as β reaches toward infinity, then the line simply returns a "stair" (not a curvy line anymore which the segment will return vertical). Such phenomenon is not what should be expected which there is barely any chance that a "vertical" line turns out to be the best fit toward the training dataset.

With this in mind, the possible values of β are limited to a certain constraint which allows the algorithm to determine whether which β would be the best fit toward the training dataset. In order to determine the value of β, the following statement is defined:

$$\beta = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - y_i^{'})^2$$
$$y_i^{'} = \frac{255}{1 + e^{-\beta(x_i - 127.5)}}$$
$$\beta \in (0, 0.1]$$

By examining the statement above, the main goal is to reduce the total loss (or MSE) of the ML model as much as possible, so that the curve generated is considered the best fit. The domain of β is chosen based on the observation of the graph under $x \in [0, 255]$ and $y \in [0, 255]$ whereas N stands for the total number of data points from the training dataset, y_i represents the provided training "labels" (remember, this is a supervised learning), and $y_i^{'}$ represents the predicted output value based on the tested β.

By spanning through $\beta \in (0, 0.1]$ with the increment of 0.001, the best-fit ML model may be determined in which such β has the lowest loss (or MSE) toward the training data as provided. With such approach, it allows the ML model to be established more accurately whereas three individual models will be established respectively to Red, Green, and Blue. For instance, the following plots represent how the training dataset (correspond to the left half of the photo) helps the algorithm to determine the best-fit curve.

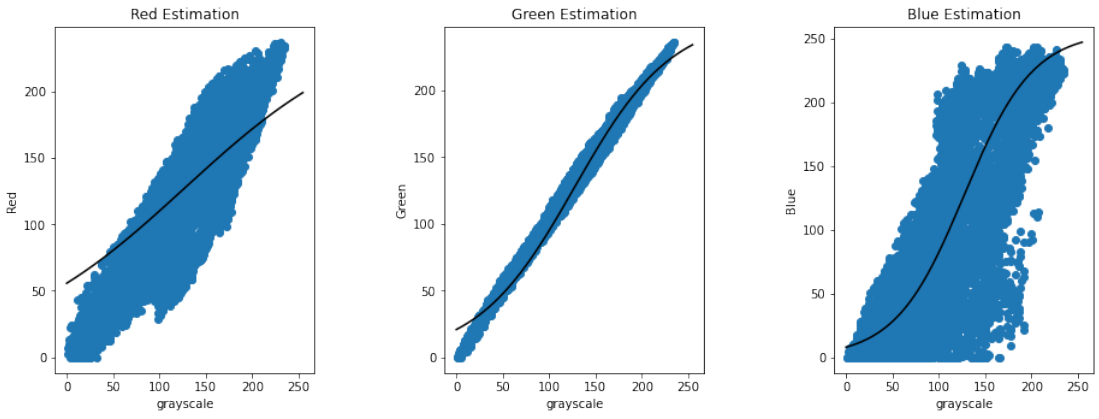

Figure 4: Values of R, G, B, With Logistic Regression Curve

In the figure above, the grayscale value to the corresponding R, G, B values are graphed in three scatter plots. The logistic regression curve is plotted as well. From the plot, we can see that the obtained best-fit curve is chosen to fit most of the data points. Especially for green and blue, the curve seems to fit the data very well. This may be due to the fact that the original image has a lot of blue and green colors.

Comparing the Improved and Basic Agents

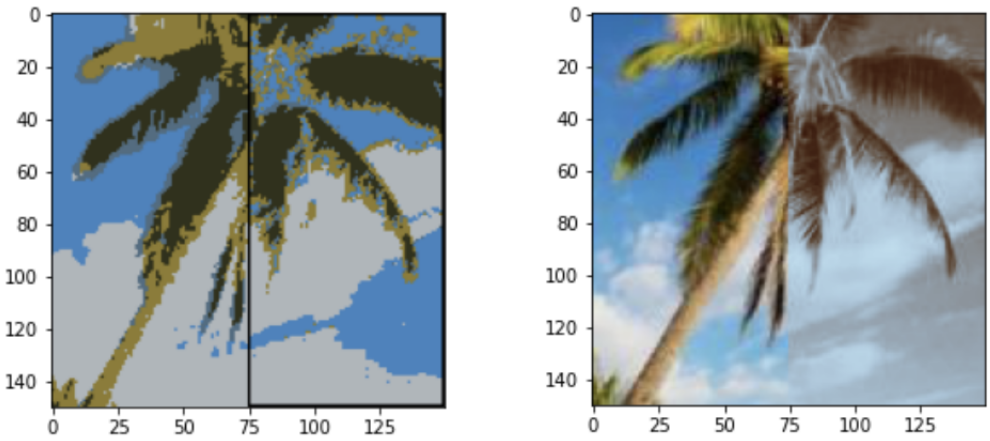

Figure 5: (left) Final Result of Basic Agent, (right) Final Result of Improved Agent

Visually, there is a significant difference between the two predicted colorized images. The basic image seems more colorful due to the 5 best represented colors, but the details of the image is lost and the objects look choppy. The improved image is not as colorful - looking like the grayscale image but with a touch of color. However, the details of the objects, including the leaves are very clear. Mean square error for basic is around R: 1789, G: 2830, B: 1005, and the MSE for improved is around R: 1135, G: 42, B: 1909. Mathematically speaking, the predictions for the logistic regression model is much better than the KNN model. Although the MSE value for the color blue is greater in the improved, the MSE for the other two colors are significantly lower. From this, we can conclude that even though one model may produce colors that are closer to the actual, too much data is lost when doing the K-Means Clustering to make up for it.

Although it is in fact that the improved ML model does not go through any data pre-processing such as color clustering, the efficiency and the quality of such algorithm is actually quite admiring. Although there may be other learning algorithm that may significantly reduce the MSE/loss (FYI, this task is to output 3 labels from an input that contain only 1 feature...), the mathematical model seems to be really concise whereas the input/output numerical values are limited. Due to this reason, this has allowed the algorithm to complete learning such training dataset within a very short amount of time verses the basic ML model. Even though the loss of the improved ML model seems to be lower comparing to the basic method from the numerical aspect, it is indeed that the improve ML photo may not look as admiring as it should have been from human's eyes, despite the fact that it keeps more information of the photo.

Bonus: Logistic Regression/OLS through "scikit-learn"

Since it is 21st century and there has been a lot of machine learning models that can be adopted. With this in mind, for the bonus ML toward the photo, the LogisticRegression and the Linear-Regression function is applied through Python's imported library of "sklearn.linear_model". Such function/package allows the user to train a ML model by simply inputting the training dataset (with labels), and the model will be ready to be tested through the inputs of testing dataset (without labels whereas the model will output the predicted labels).

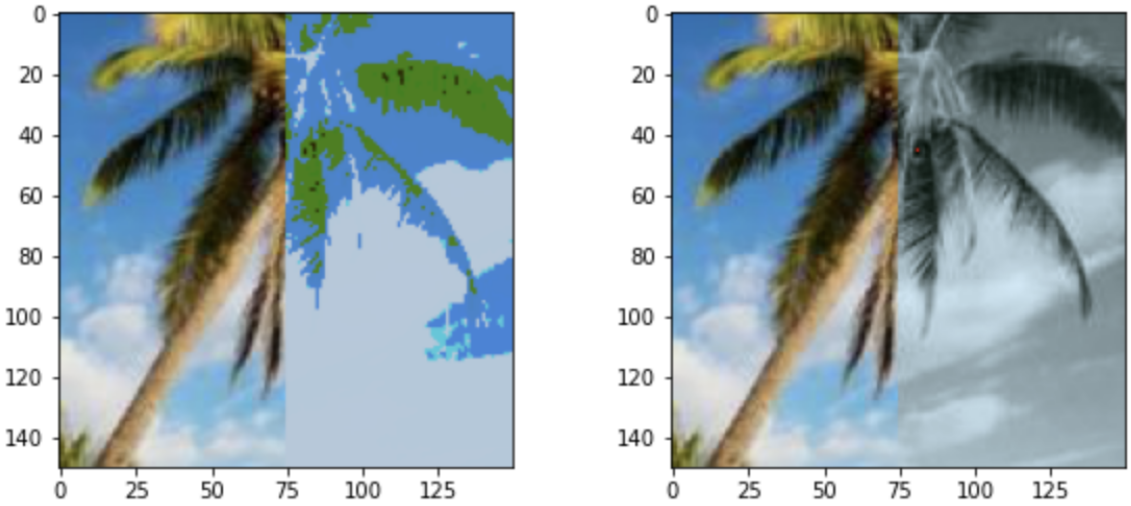

Figure 6: (left) Sklearn Logistic Regression Model, (right) Sklearn Ordinary Least Squares Model

Visually, the logistic regression model from sklearn obtained bad results when predicting the r, g, b values. The colors are a bit brighter like the basic agent but the majority of the details are lost from the tree. On the other hand the OLS Model from sklearn looks similar to the predictions of the improved agent. Overall, the left looks like the basic agent and the right looks like the improved agent.

Mathematically, the MSE for sklearn regression is about R: 2004, G: 1178, B: 5031, and the MSE for sklearn OLS is about R: 813, G: 26, B: 2019. Bringing in the MSE for basic agent at R: 1789, G: 2830, B: 1005, and MSE for improved agent at R: 1135, G: 42, B: 1909, we can see that the model with the least MSE is the sklearn OLS model. Similar to our advanced agent, the error for the color blue is very high, probably due to the large amount of blue in the original image, but the error for the other two colors are significantly lower. On the other hand, sklearn regression actually produced the worst results with the most error out of the four models.

In conclusion, it is quite interesting of how a grayscaled photo can be reconstructed back to a photo that consists of RGB property based on the training dataset toward ML. According to the numerical comparisons of MSE and loss above, there are various methods/mathematical models that attempt to minimize such error. Although it has been the most objective aspect to compare/observe the error between the actual data labels and the predicted data outputs, RGB in fact has a "magic" combination whereas the photo that contains the least MSE may not be the most admiring from human's beauty aspect. Indeed, it is only concluded that the photo with the lowest error may contain the most information which may leave future editors enough room to modify or adjust the photo if needed.