



CS440

# Colorization

**By Maitri Shah, Kajol Bhat & Ashni Patel**

INTRO TO AI SPRING 2021

# Contents

I	Part One	
<b>1</b>	<b>Questions and Write Up</b>	<b>3</b>
1.1	<b>Basic Agent</b>	<b>3</b>
1.1.1	K-Means Algorithm	3
1.1.2	Pixel Patches	4
1.1.3	Final Result	4
1.2	<b>Advanced Agent</b>	<b>5</b>
1.3	<b>Comparisons</b>	<b>6</b>
1.4	<b>Conclusion</b>	<b>7</b>
<b>2</b>	<b>Bonuses</b>	<b>8</b>
2.1	LaTeXed Report	8
2.2	Clever Acronym	8
2.3	Why is a linear model a bad idea?	8
2.4	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?	8
2.5	Division of Work	9
2.6	Honor Statements	9



# Part One

<b>1</b>	<b>Questions and Write Up .....</b>	<b>3</b>
1.1	Basic Agent	
1.2	Advanced Agent	
1.3	Comparisons	
1.4	Conclusion	
<b>2</b>	<b>Bonuses .....</b>	<b>8</b>
2.1	LaTeXed Report	
2.2	Clever Acronym	
2.3	Why is a linear model a bad idea?	
2.4	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?	
2.5	Division of Work	
2.6	Honor Statements	

# 1. Questions and Write Up

## 1.1 Basic Agent

The following image was used for this project. In order to pre-process the data, we converted the image to a gray scale and stores it into one array, and then also stored all of the individual rgb values of the original photo into a different array.



This image was then converted to black and white and then recolored using the below algorithms.

### 1.1.1 K-Means Algorithm

For our k-means algorithm, we started by picking five random coordinates. We then used the following equation to calculate the Euclidean distance:

$$Distance(x,y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1.1)$$

The closest euclidean distance was then used to group each pixel by the best representative colour. Each color group was entered into a dictionary. The keys corresponded to the color group and the values were the coordinates that were clustered into that respective color group. The left side of the image was then, recolored based on the k-means clustering algorithm. For each pixel in the left half of the color image, the true color would be replaced with the nearest representative color from the clustering. In this case, our clustering chose 5 colors but if we increase the value of k, the clarity of the image would increase accordingly.

### 1.1.2 Pixel Patches

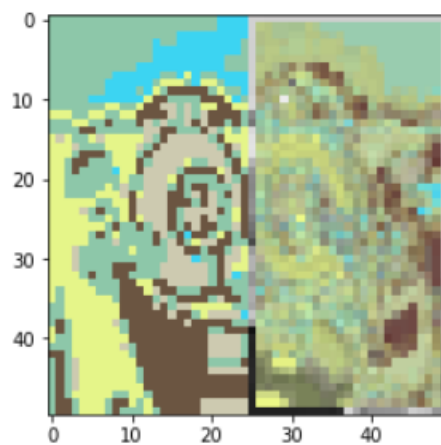
Once the pixel values of the left side is determined. To recolor the right side, which is still black and white, we created a series of 3x3 pixels for comparison. We found the six most similar 3x3 gray scale pixel patches in the training data. Our training data is the left side of the black and white image. Each 3x3 set of pixels on the right side was compared with six of the most similar grey patches on the left side. The similarity between the patches is calculated using a nine dimensional distance formula to determine the Euclidean distance.

We use the color representative of the re-colored middle pixel for each of the six specified patches. After identifying the six most similar patches, we examined the representative color of these patches and recolored the right side pixel based on the plurality color.

### 1.1.3 Final Result

We obtained the final result by training our model using K-means Clustering. As can be seen with the image below, the agent is able to color the left half side, and right half side relatively well using the method. The right side appears to be based off of the clusters that were created on the left side. However, due to the limitations of our computer, we were only able to run our code with a 50x50 picture, but the output would likely have been better if we used a higher pixel count. This is as our image would also be more defined and accurate if we used a bigger dimension for the picture as we would have more pixels to work with and a mishap with one pixel wouldn't affect the final result as much.

<matplotlib.image.AxesImage at 0x1218a4df0



Basic Agent Spongebob

Below, we have also included a copy of an image that we ran that was 100x100 in size for the left side, and from this it is clear that the final input's clarity would have looked more similar to this one if we had used a larger image. However, based on the limitations of our computers, we had to switch to a lower image size for our project.

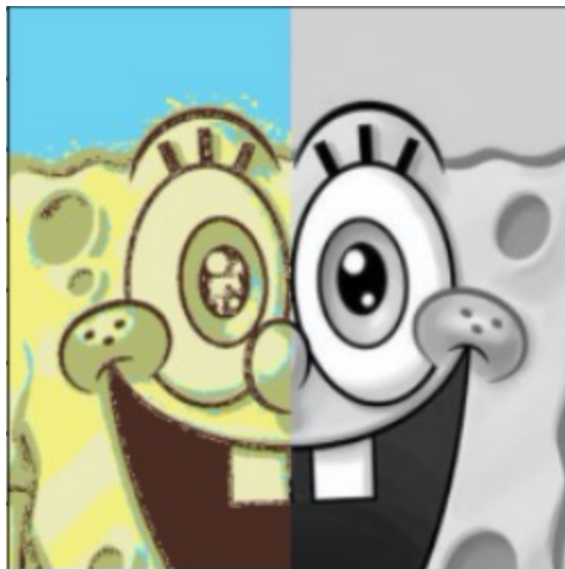


Image of left side with larger pixel number

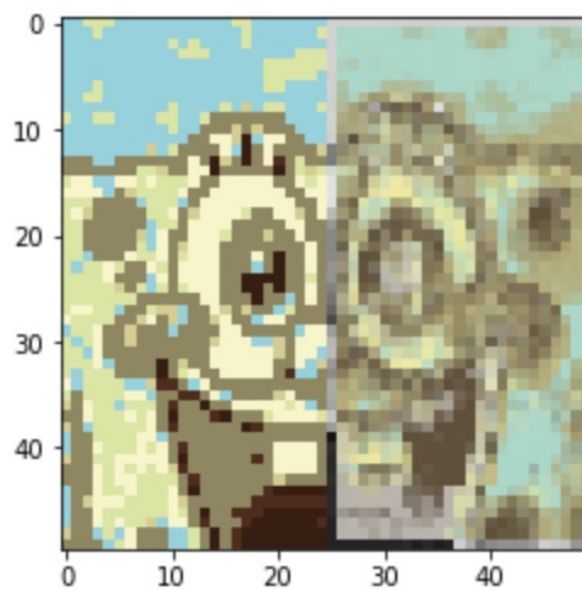
## 1.2 Advanced Agent

For our advanced agent, we decided to use the following idea : "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."

For this, we first took our basic k-NN algorithm and decided to make it more accurate by increasing our k value to 7. After experimenting with the k values, we found that the most accurate images showed us when k was 7. To improve our agent, we then decided to think of the k-NN cluster algorithm as a weighted regression problem. We made a function to calculate the weights that we would multiply each of the euclidean distances by multiplying it by  $1/(distance^2)$ , so that the values that were closest to each cluster would have the highest priority when we were selecting the top 7 clusters. The additional aspect of weight would allow the function to minimize errors. After this is done, we then try to predict the value that will be at the center of each patch by taking the average of the grey scale values in the surrounding cells. Using this, and the weighted number, we add this into our list and use it to predict what the most likely value for that given pixel would be.

Overall, we are quite confident that our algorithm would have performed better had our computers been able to handle a 100x100 image, and we believe that the quality of the image is likely due to the fact that it was done using a 50x50. Furthermore, if we had extra time, we could have tested this by implementing other methods and comparing our results, such as comparing it to the results of an implementation using the sigmoid function. Also, to get the predicted values, we used an average function, which in general worked, but sometimes made the resulting color looks slightly different. For example, in the spongebob image, the average of a patch resulted in teal-blue on the right side rather than the light blue that is seen on the left. If we had more time, we would have instead either made this a majority function to make these colors match better.

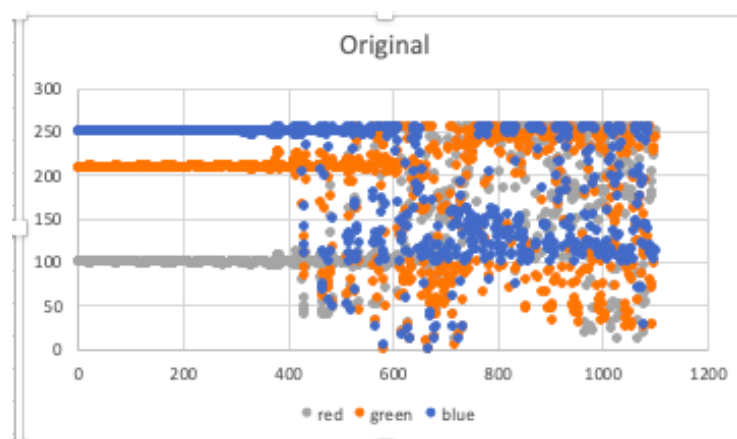




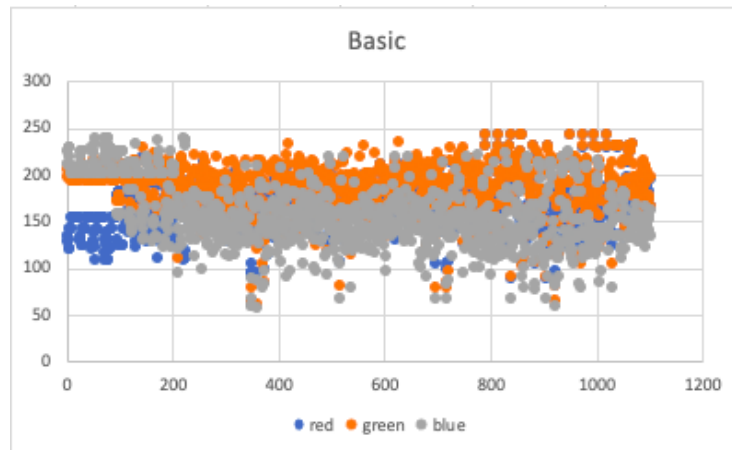
Advanced Agent Result

### 1.3 Comparisons

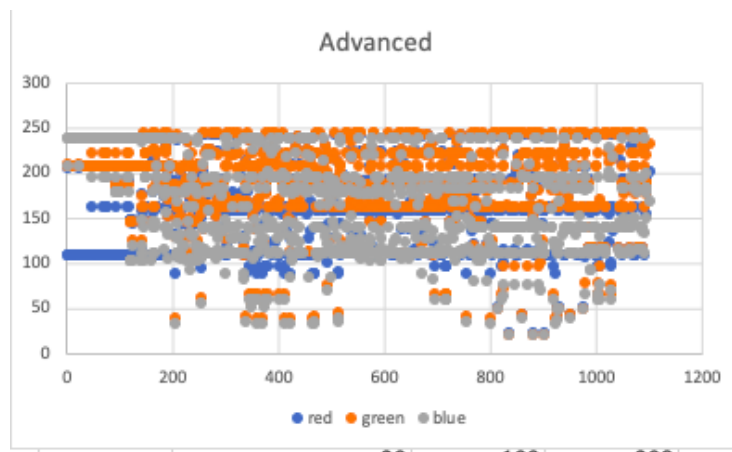
In general, it appears that our advanced agent is working better than the basic one. In order to compare this, we outputted the predicted values of each of the patches (for basic and advanced) that we went through to the actual values that can be seen in the graphs below. The y axis shows the rgb numeric value for each of the colors listed.



Original Colors



Basic Colors




Advanced Agent Colors

As it can be seen, in terms of shape of the graph, the advanced agent is a better fit than the basic agent. While it is not a perfect match, we can see that the shape in general matches it more and there seems to be less variability between the colors. We can see this numerically as the graphs show that the distance between the average rgb value for the advanced agent appears to be smaller than that of the basic agent. Also, based on the pictures themselves, we can see that the colorization for the advanced agent appears to work better, as the image of the cartoon is a lot more clear in the advanced agent's final output.

## 1.4 Conclusion

If we were given enough time, computational power, energy and resources we would have improved our advanced agent by using a neural network. The neural network would likely not result in an image that would be as pixelated because we would be able to use a larger image. We would input our patches onto the neural network and then compare the output of the neural network with the actual values from the original image in order to quantify the effectiveness of the image. We would also repeat the process for multiple iterations to ensure that the neural network was as accurate as possible as was able to learn as it continued. In order to account for over fitting, we would ensure that there were enough clustering but not too many clusters to the point that the data would be over-clustered.





## 2. Bonuses

### 2.1 LaTeXed Report

This entire report was written using LaTeX

### 2.2 Clever Acronym

BAMBI - Basic Agent Mediocre Because not Improved

CAMI - Colorization Agent More Improved

### 2.3 Why is a linear model a bad idea?

In general, it is a better idea for this project to use a parametric model rather than a linear one. One of the reasons for this, is that in a linear model, you must assume that the data will be relatively consistent, with very few outliers and changes in patterns. However, this would not always be the case as the consistency of a given photograph may be variables, with many different colors and patterns. Furthermore, in a linear model, there is the assumption that insignificant values can simply be removed like in the case of outliers, but for this project where we have to color a photograph, values cannot be removed and must all be included. Therefore, we cannot use a linear model for this project.

### 2.4 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?

Generally, the ideal  $k$  value will be based on which  $k$  value will result in the smallest error when clustering all of the values in the data set to their respective  $k$  clusters. This will have to be done through a series of trials. However, generally speaking a larger  $k$  values will be better. For our project, we tested different values of  $k$  between 5 and 10 and found 7 to be the most ideal. This

could also be done using the elbow method, a formula that runs k-means on all possible clusters and determines which value would be optimal for the data.

## 2.5 Division of Work

All of the work was done over Zoom calls as a group so all of the work was split evenly (Acknowledged by Maitri Shah, Kajol Bhat, and Ashni Patel)

## 2.6 Honor Statements

**Maitri Shah (ms2804):** On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.

**Kajol Bhat (kkb56):** On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.

**Ashni Patel (agp96):** On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.