

Introduction to Artificial Intelligence

Project 4: Colorization

Brenton Bongcaron and Abe Vitangcol
NetIDs: bdb101 and alv88

May 5, 2021

We have read and abided by the rules laid out in Canvas, We have not used anyone else's work for our project, and our work is only our own.

Contents

1	Introduction	2
2	The Image	2
3	The Basic Agent	2
4	The Advanced Agent	3
5	Contributions	5

List of Figures

1	Wow, such beauty, much colors, many fun. Will it go to the moon?	2
2	Colors Much, Beauty Such, Woah.	3
3	Wow, such beauty, much shading, many wows.	5

1 Introduction

Imagine this: you want to send an image to someone, but trying to send the image will cause some error, saying how the image was too large to send. This is commonly seen on applications such as Discord where they limit the size of the images people are allowed to send (unless, with paid perks of course) and the image cannot send. So, instead, the image gets rejected to send unless it has been compressed to lower the size of the image file. So, then the question becomes: how does the image become compressed and the file size becomes lower than the original? The answer: through simplifying the image into similar colors, thus reducing the size. Hence, this project will do such thing, simplifying an image and use less colors to represent the same thing.

2 The Image

To test out our algorithms to see how well it does, we have to use a common image and see the algorithm's quality in terms of both representation (how well it represents the original image) and in terms of aesthetics (how appealing it looks to the eye). The image needs to have a balance of colors on both sides and not too much of a color spread. The image we needed was, in fact, the doge, and thus, the doge became the image we tested on.

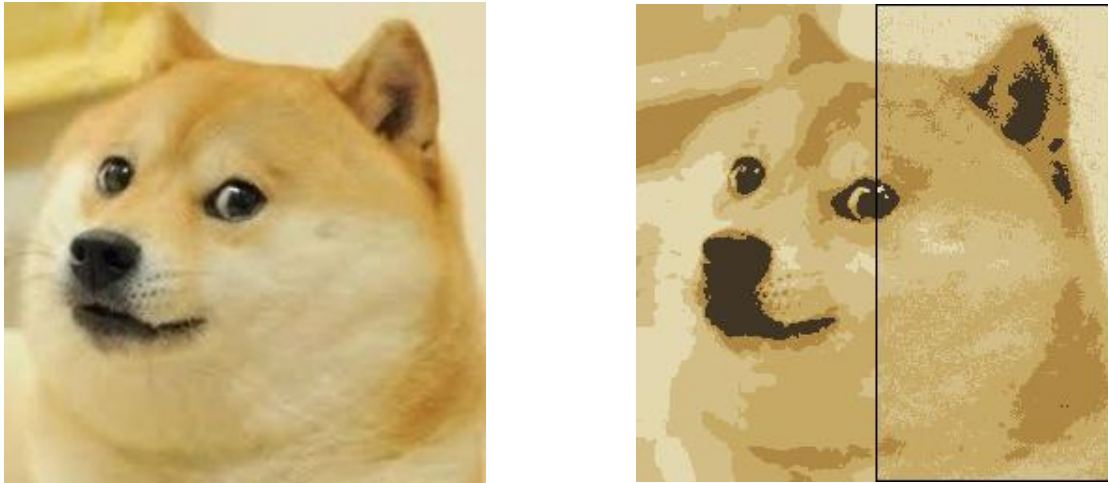


Figure 1: Wow, such beauty, much colors, many fun. Will it go to the moon?

3 The Basic Agent

To start, the basic agent was necessary as a baseline of how well our advanced algorithm works. The basic agent begins with clustering the colors into 5 best representative colors using k-Nearest Neighbors calculations. Then, the basic agent has 2 arrays, one a grayscale version of the doge, the other the original image. Using the left half of the grayscale image as training, it replaces 3x3 patches with one of the representative colors accordingly. Then it goes to the right half and tries to fill in the rest of the colors based on what it learned through the training, with a black border

around the right half because of how some tiles do not give enough information to complete the image. The result is the following:



Original

Basic Agent

Figure 2: Colors Much, Beauty Such, Woah.

4 The Advanced Agent

The newest question then becomes: can we create something that beats a 5-Nearest Neighbor calculation and create a better image than the basic agent? The answer was a simple one, there always was a way to beat it, specifically, using logistic regression. We do this using a method called in Al.py, named `weightFitting`:

```
def weightFitting(image):
    im_width, im_height <- image.size
    rgbMatrix <- numpy matrix form of image
    grayIm <- numpy matrix form of grayscale image
    alpha = 0.0001
    wR <- initialization of all weights for "red" model to 0.5
    wG <- initialization of all weights for "blue" model to 0.5
    wB <- initialization of all weights for "green" model to 0.5
    for 150000 rounds of training:
        randPixel <- random pixel on LEFT side of image
        r, g, b <- RGB values of randPixel
        gray = [1]
        for x in range(-1,2):
            for y in range(-1,2):
                currentPixel <- (randPixel[0]+x, randPixel[1]+y)
                gray.append(grayscale value, divded by 255, of currentPixel)
    # Note that dotP() means dot product
    Rx = dotP(wR, gray)
    Gx = dotP(wG, gray)
```

Bx = dotP(wB, gray)

predR = 255.0 / (1 + exp(-1 * Rx))

Do the above calculation for the other two colors

gLossR = [(predR - r)*predR*(1 - predR/255.0)*gray[i] for i in range(len(gray))]

Do the above calculation for the other two colors

update the weights of the colors' models

return the weights of the colors' models

The loss function is defined as:

$$\text{Loss} = (r - 255\text{sigmoid}(Rx))^2$$

And sigmoid is defined as:

$$\text{sigmoid} = \frac{1}{1 + e^{-x}}$$

$$\text{sigmoid}' \text{ (Derivative of sigmoid)} = \text{sigmoid} * (1 - \text{sigmoid}) \text{ or } -\frac{e^{-x}}{(1 + e^{-x})^2}$$

By plugging in the sigmoid into the defined loss function and taking the derivative:

$$\text{Loss}' = 2(r - 255\text{sigmoid}(Rx))(-255)\text{sigmoid}'(Rx)R'x$$

Rx is the dot product between the weights of a color model (red, blue, green) and the grayscale, and the derivative of a dot product is:

$$R'x = \frac{d(\text{grayscale})}{dx} \text{weights} + \frac{d(\text{weights})}{dx} \text{grayscale}$$

But with the dot product, taking the derivative of the dot product with respect to x causes R'x to simply return $w_x g_x$ (with w meaning weight and g meaning grayscale value), so, we can plug this in:

$$\text{Loss}' = 2(r - 255\text{sigmoid}(Rx))(-255)\text{sigmoid}'(Rx)w_x g_x$$

Note that the gradient of the loss function is calculated by the formula:

$$(\text{predicted} - \text{actual}) * (\text{predicted}) * (1 - \frac{\text{predicted}}{255.0}) * \text{grayscale value}$$

Which was shown in doing the above calculations and by plugging in the values we were given. The predicted color value(for each color) is:

$$R = \frac{255.0}{(1 + e^{\text{dotP}(wR, \text{gray})})}$$

$$G = \frac{255.0}{(1 + e^{\text{dotP}(wG, \text{gray})})}$$

$$B = \frac{255.0}{(1 + e^{\text{dotP}(wB, \text{gray})})}$$

where $\text{dotP}()$ denotes dot product. Note the Color Sum where it says "Rx", "Gx", and "Bx" in the code. After doing all of this, one update is done on all of the weighted values of the colors, which is where we use alpha and the loss values:

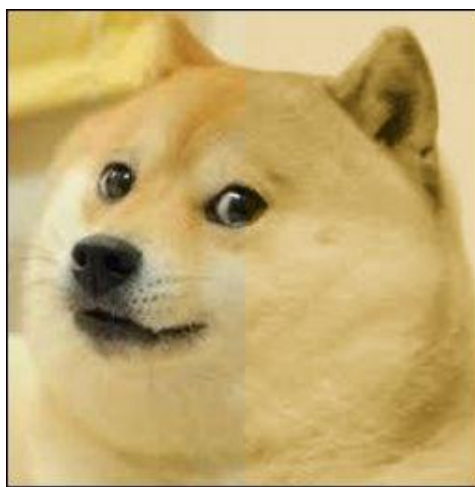
$$(\text{Current Color Weight}) - (\alpha * \text{Gradient of Loss of Color})$$

This is what is used to update the weights of the colors and is returned after all the trials. Note the alpha and initial weight values seen within the code were found by performing many tests and checking various values before settling on the current values seen. As for the training, more was better since the weights were going to converge anyway and the training was fast, so more was done. This concludes the function `weightFitting`, with the weights of the colors returned to the advanced agent where it uses them to complete the picture.

The result of the Advanced AI is shown in Figure 3.



Original



Advanced Agent

Figure 3: Wow, such beauty, much shading, many wows.

In terms of quality, it definitely has more colors than the basic agent and, aesthetically speaking, it looks much better than the, err... abstract art doge. The advanced agent colors and shades the doge really well, loses the black border surrounding the right-half of the picture, and, in terms of time, it completes the doge faster than the basic agent. The basic agent takes around 7 minutes to complete its picture whereas the advanced agent took only 30 seconds to a minute to complete with a better quality. Given how both used the same picture and the way both agents completed the doge, it seemed fair to say the basic agent was not as capable as the advanced agent, but both did their jobs well. With how well the advanced agent does, it was given the name **Stronk Vectoral Weightroom**, or **SVW**.

However, should there be enough time and resources to complete this (and not a span of two to three weeks because of unfortunate college classes), there would be plans to use neural networks to color the image. There would also be plans to make the algorithm faster (faster than 30 seconds because there is never anything wrong with faster) and test it on more than just doge.

5 Contributions

Brenton Bongcaron [bdb101], Section 3

I did all of the coding of the basic and advanced agents. I also made edits to the \LaTeX document.

Abe Vitangcol [alv88], Section 8

I completed this \LaTeX document and did some proofreading and commenting on the code. Wording the \LaTeX document was something both my partner and I agreed I liked doing as well as having fun with the doge memes here. Unfortunately, many things came up on my end (mainly affecting my mental health) and thus I wasn't able to work on this project as much as I wanted to. Many apologies to my partner for being unable to help well on this one.