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

List of Figures

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 verison 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:

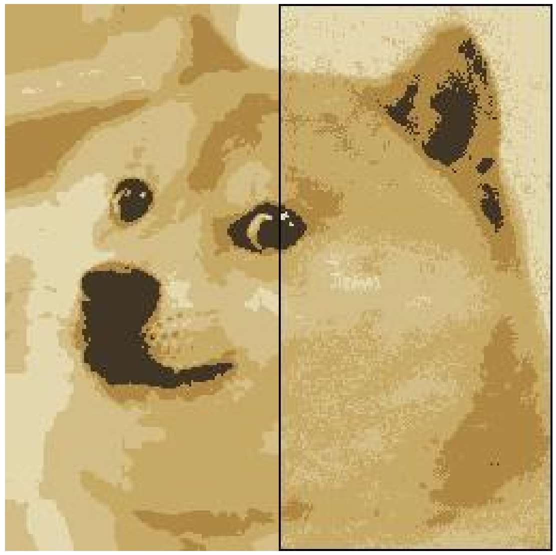

Figure 2: Colors Much, Beauty Such, Woah. Lowkey deepfried doge.

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 AI.py, named weightFitting:

```
def weightFitting(image):
    variables height and width are set equal to the size of the image
    make a variable an array representation of the image
    make a variable an array representation of the grayscale image
    alpha = 0.0001
    Weight of Red, Blue, and Green equal to an array with 10 slots, each with 0.5 value.
    for 150000 rounds of training:
        select a random pixel
        select random values for r,g,b
        gray = [1]
        for one row below and above:
            for one column behind and ahead:
                append the random pixel area into gray

        Rx = sum of weightedRed * gray for all values in the weightedRed
        Gx = sum of weightedGreen * gray for all values in the weightedGreen
        Bx = sum of weightedBlue * gray for all values in the weightedBlue

        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 weighted values of each color

 return the weights of the colors
```

Note that the loss function is calculated by the formula:

$$\frac{(\text{Pred Color Value} - \text{Real Color Value}) * (PredColorValue) * (1 - \text{Pred Color Value})}{255.0} * \text{Gray Array Value}$$

And it is done for every value in the gray array. The predicted color value is done by:

$$\frac{255.0}{(1 - e^{\text{-Color Sum}})}$$

Note the Color Sum where is 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:

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

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 was going 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 the following:

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 (basically a beige variant of black-and-white), loses the black border surrounding the right-half of the picture, and, in terms of time, it complete 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. Basic agent had brighter colors but lost the quality in terms of per-pixel detail whereas the advanced had most of the shading but lost the vibrant colors in a significantly faster time.

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 the same method but add in the vibrant colors. 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

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.