AI Assignment 2

AI Art - 13 April 2021

Introduction -

The main aim of this task was to implement some efficient genetic algorithm, in order to *distort* some pixels of the picture to let the computer make art based on certain parameters. This art can be done using *circles*, *triangles*, etc basically any geometric figure or even images like *minions*, *electrical components*, etc.

In my case, I have used *circles*. My artwork is populated with small circles filling the pixels and developing a clone of the input image with *RGB* (Red-Green-Blue) circles.

Note - Image size is taken as 512*512 (If it exceeds then I have given the function of resizer to change it to 512x512)

Art -

“Art is a diverse range of human activities in creating visual, auditory or performing artifacts (artworks), expressing the author's imaginative, conceptual ideas, or technical skill, intended to be appreciated for their beauty or emotional power”

So from my perspective, we made a set of shapes and colors that are vivid and distinct and, hence it is art.

The idea of our photo filter is to make a new image from an initial image from colored circles looking like “bubbles” in some old game.

Chromosome Representation -

I am decomposing the future output image into a *circular* pattern, and the algorithm tries to choose the most appropriate color for that specific circular block. This is done in the sense to maximize the fitness score. So, my individual in the algorithm is an inscribed circle into the square block of some color. It is represented as an array of *3 real-valued* numbers in the range [0, 255]. The description of color in RGB is the random numbers in that specified range.

As a result, the algorithm composes up all those best individuals, so that there is a new piece of art.

Fitness Function -

I calculate the mean values on each color channel over the square block of the source image. Then this score i.e.(3D distance squared) is calculated between chromosome and mean values. The overall fitness score is calculated as $f = 1/(1 + score)$.

As a result, we get values between 0 and 1. The closer the value to 1, the more appropriate color for the future images is selected.

Selection Mechanism -

The chromosomes with higher fitness scores generally have a higher probability to be chosen for the next generation. For the selection mechanism, a *roulette wheel* was picked.

Algorithm -

1. Calculating *total fitness score* of the population as the sum of all fitness scores: $\sum_{i=0}^{n} fitness(i)$. {where *n* is the number of individuals in the population, *fitness(i)* is the fitness score of *i-th* chromosome in the population}

2. Calculating *probability* of each chromosome to be selected in new generation:

   $Total = \sum_{i=0}^{n} fitness(i)$. {where *i* is in [0,n]}

3. Calculating the *cumulative probability*.

4. Generating a random real value for each chromosome, for applying the *roulette wheel* technique: *R(i)* in [0,1]. {where *i* is in [0,n]}

5. Determining which section of cumulative probability function includes *R(i)*, and respectively taking the corresponding chromosome.

6. Image with size 512x512 consist of 262144 pixels (512x512 = 262144). And we will change the number of circles on the canvas. After that, we evaluate all circles. So, we take a *circle(color, radius, location)* and compare the color of the circle with the color of points on the initial image. Take a point (x,y) and calculate average color of the polygon [(x-1, y-1), (x-1, y+1), (x+1, y+1), (x+1, y-1)] from initial image.

7. If the value of fitness function of circle color and the average color of the polygon is less than the value which we set then we make a child of this parent. The child has the same location and radius.

8. If the fitness function for the child has more than the value which we set then we replace the child with the parent and take the next circle, else we make a new child.

9. Kind of, for the dog in picture *3.png* and quality, we have 262144 circles. And we make 793612202 children. On average it has 3027 children for one parent.

10. Lastly, we plot the figure on our canvas

Crossover Function -

In my code, the number of mate chromosomes is controlled by a constant and is approximately equal to 30%. Mate chromosomes are randomly chosen.

One-cut point technique is used for implementing the crossover function. In this technique, we need to randomly choose the position in the parent chromosome and exchange all genes of another parent chromosome from that position.

Mutation Criteria -

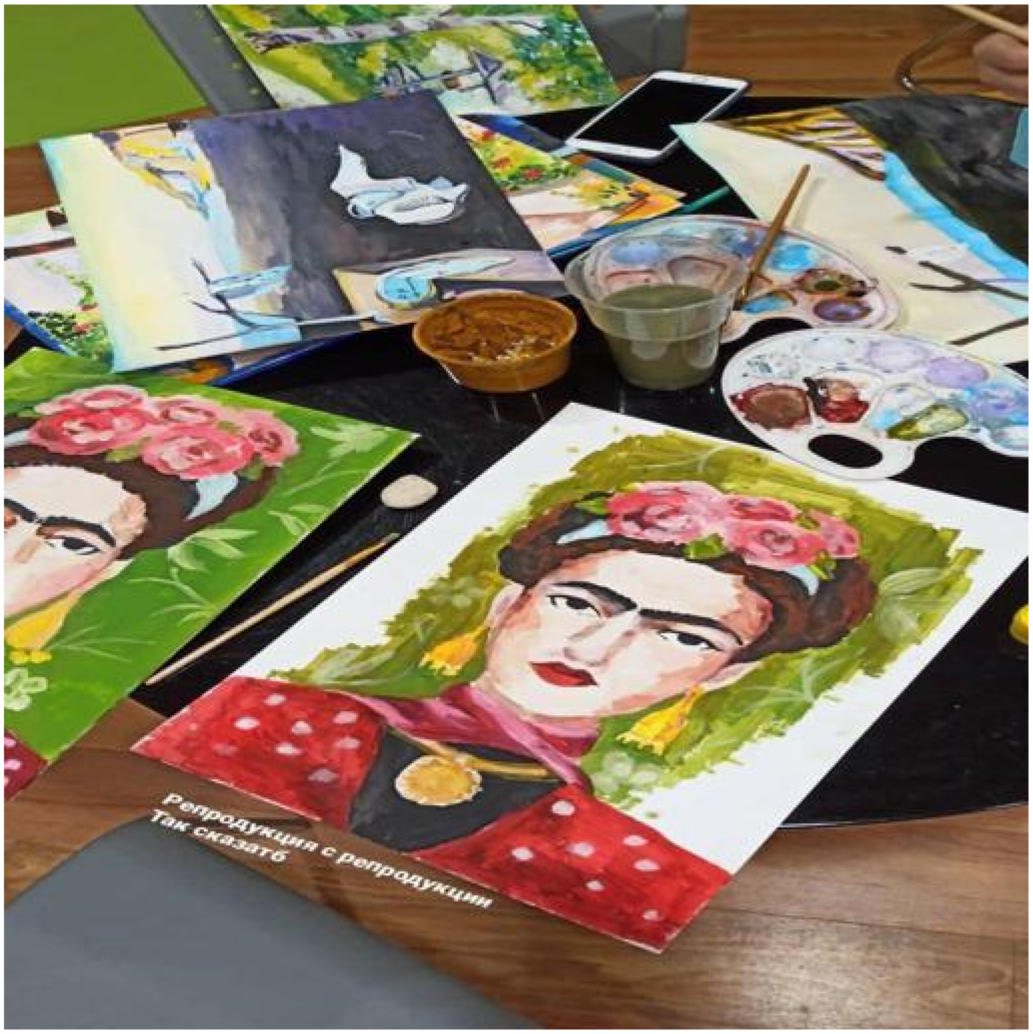
The number of mutations done in chromosomes controlled by a constant is approximately equal to 50%. This probability is so high, because of the complex RGB scheme production of color. It is harder to estimate one channel of color because the fitness function is calculating scores over 3 channels.

As a result, it’s more efficient to mutate random genes in the hope to pick a more appropriate color. Also, this way is efficient enough.

Image Manipulation -

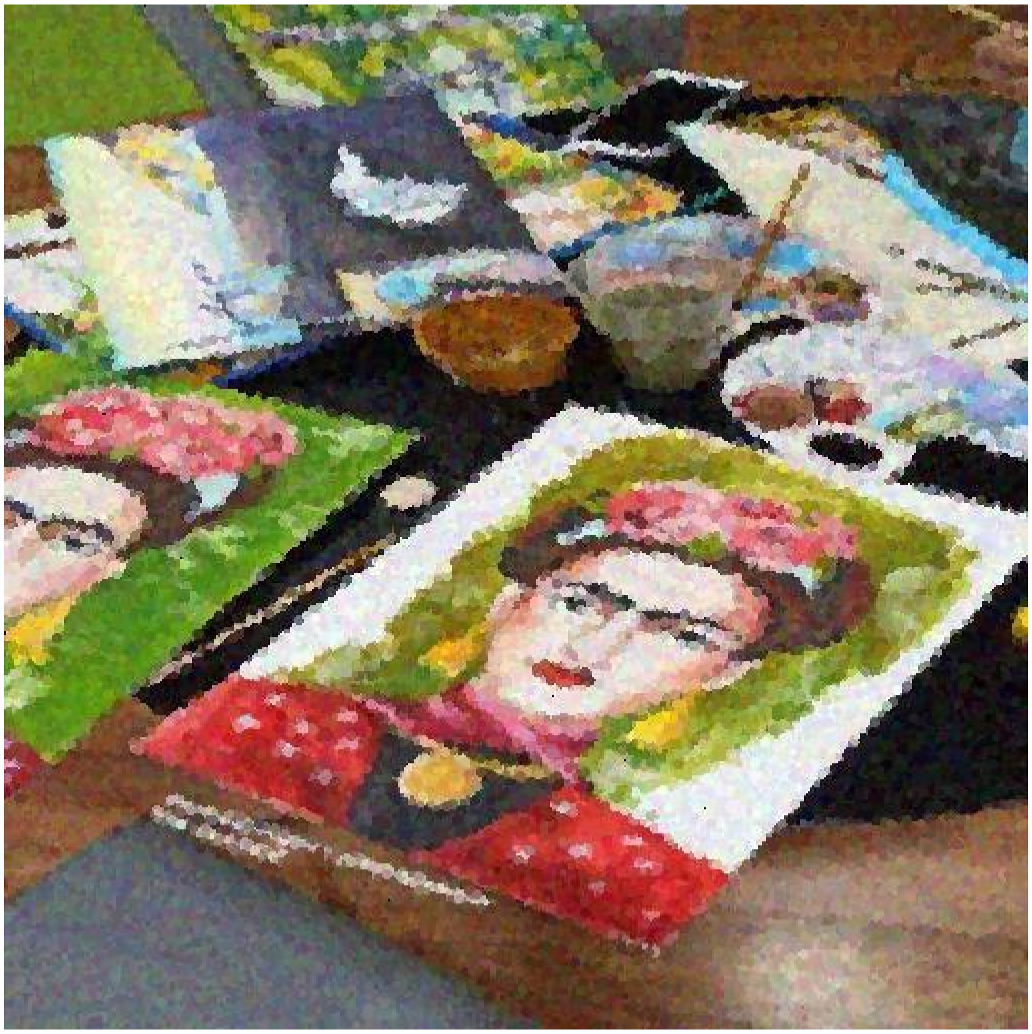
The algorithm adds padding by reflecting borders to make the image divisible by circular blocks with sides equal to *SIZE* in px.

Illustrations -

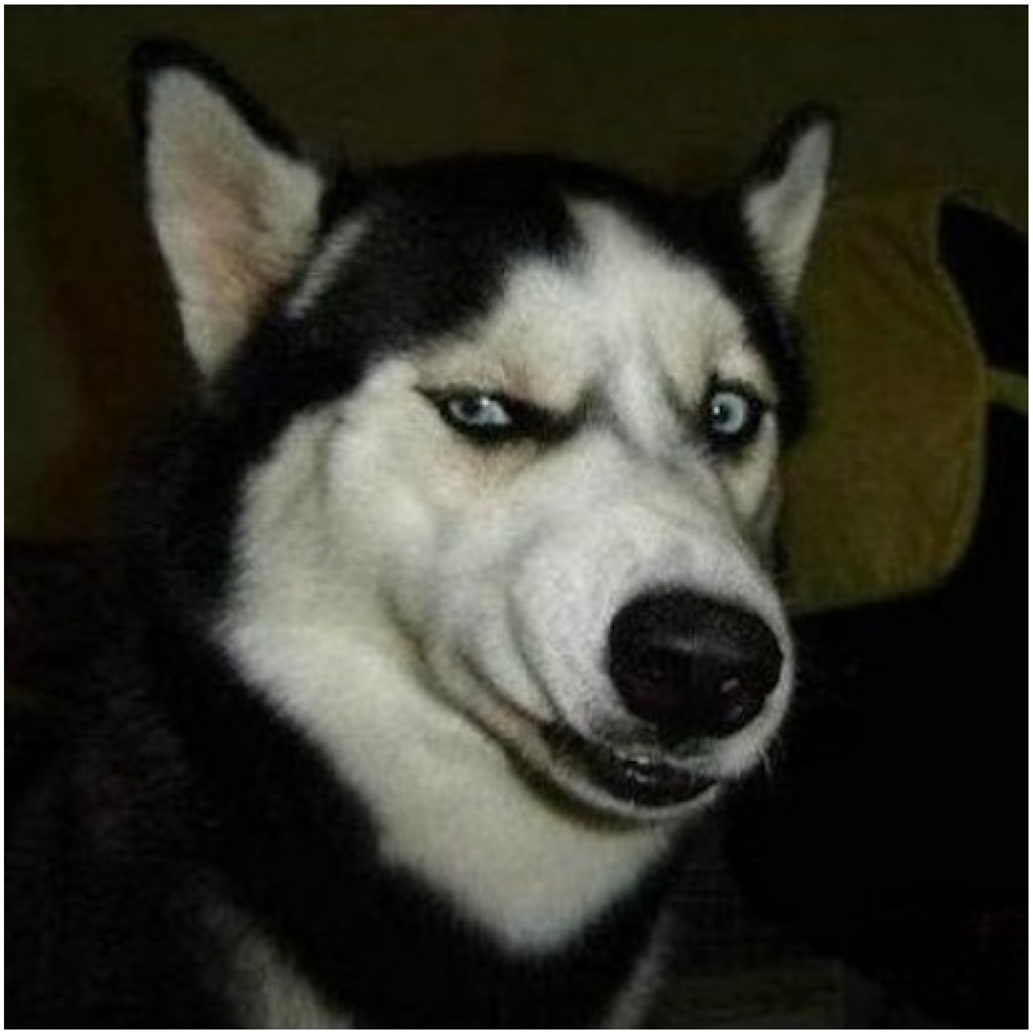


1. Input - al.jpg

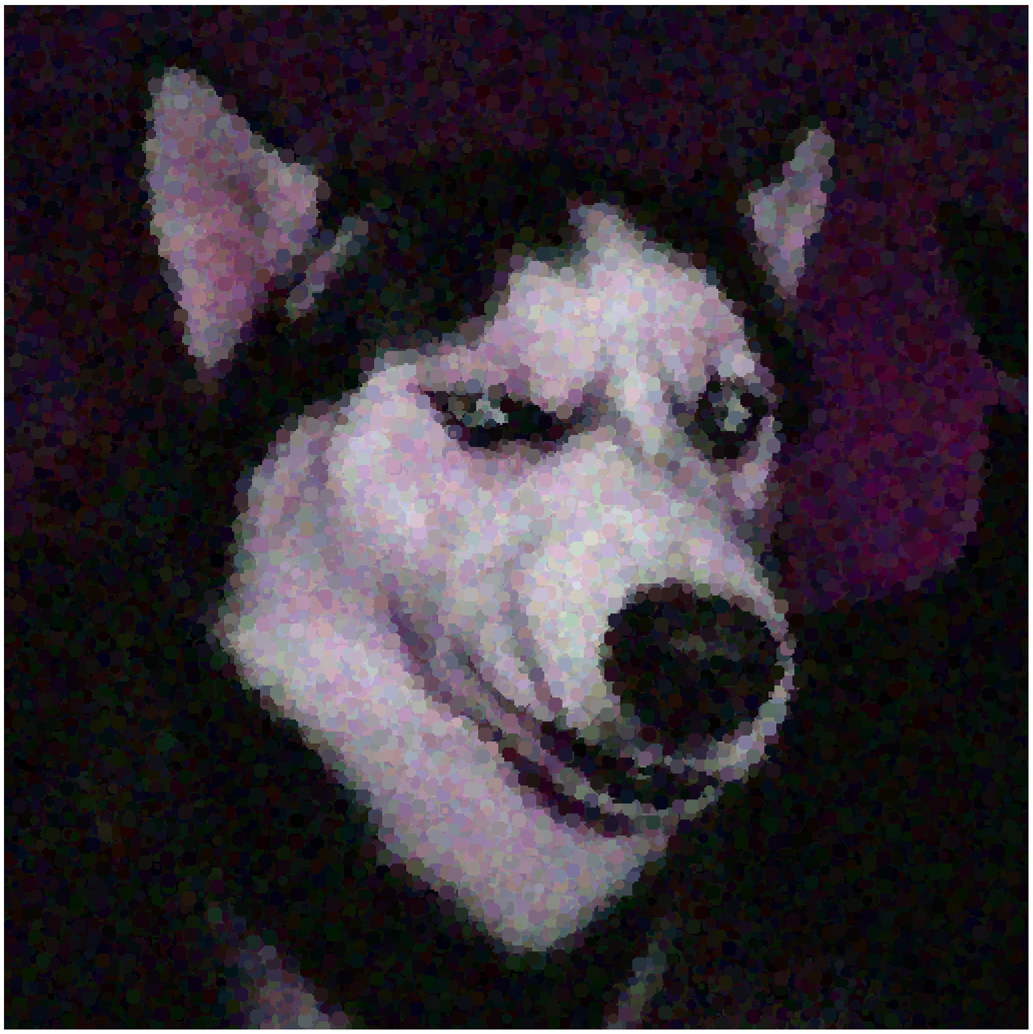
Output: Radius - 3, Quality - 95%, Time taken - 23s



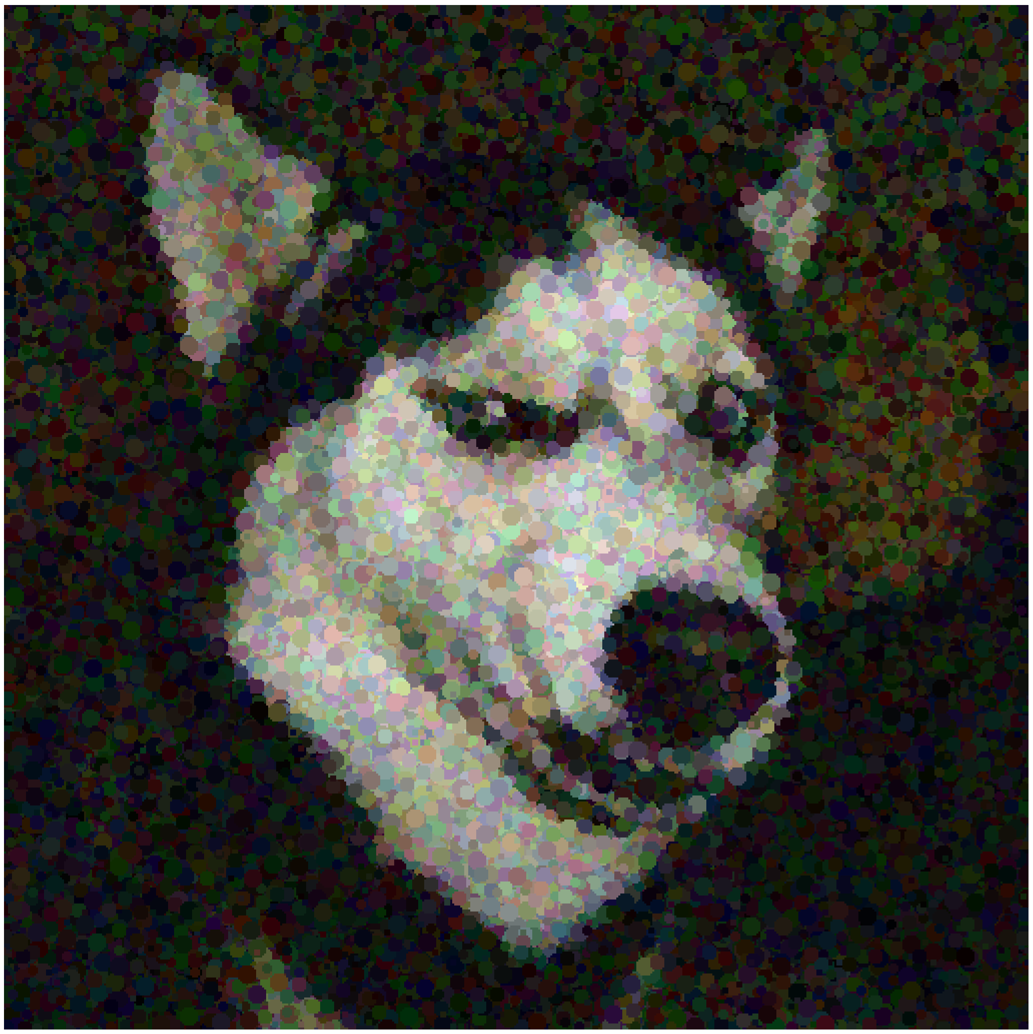
2. Input - 3.png



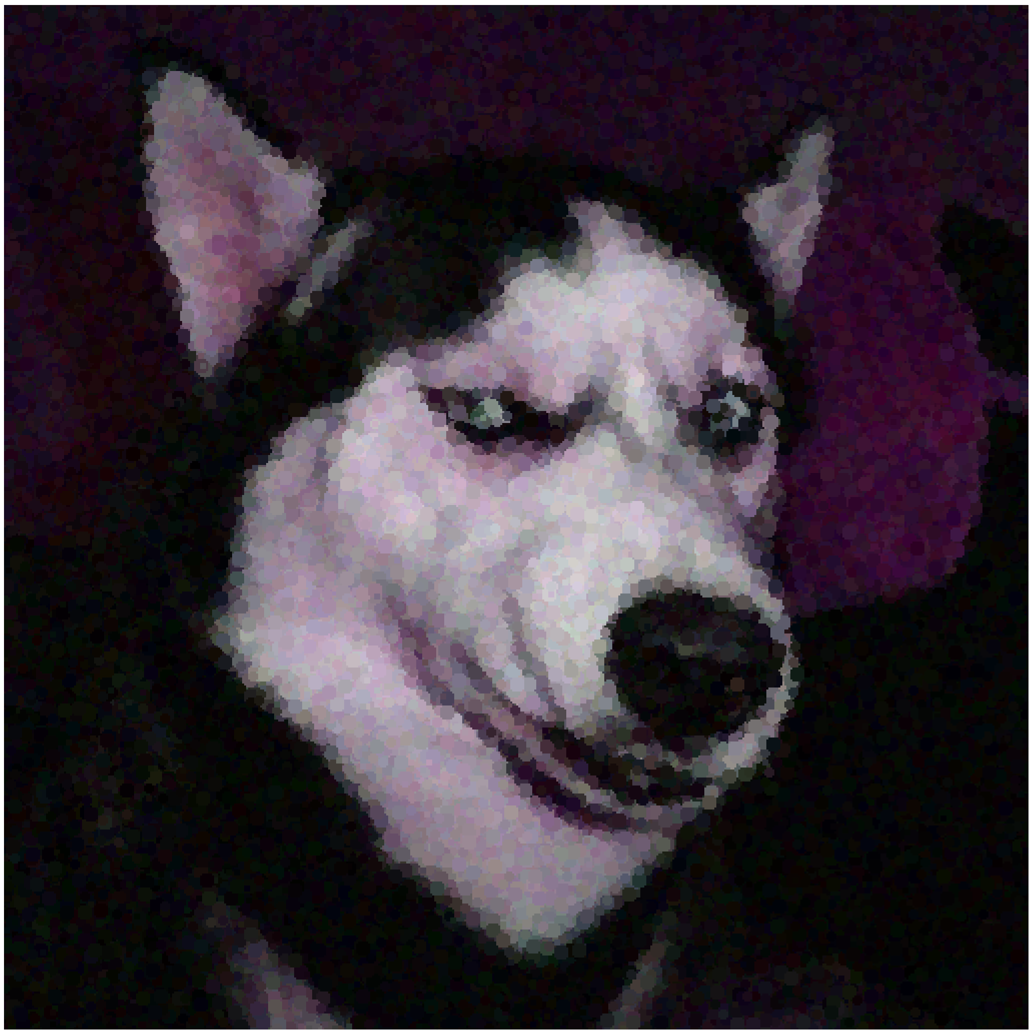
Output: Radius - 4, Quality - 95%, Time Taken - 43s



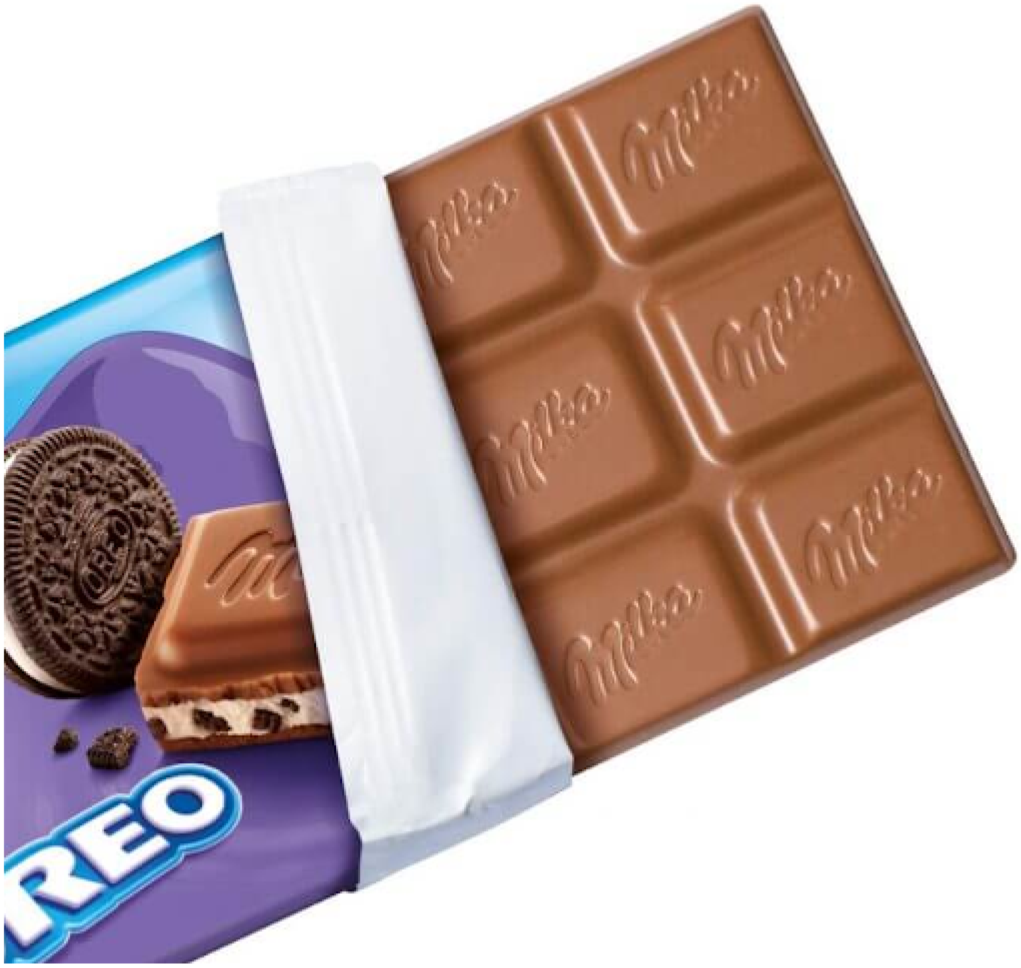
Output: Radius - 5, Quality - 90%, Time Taken - 32s



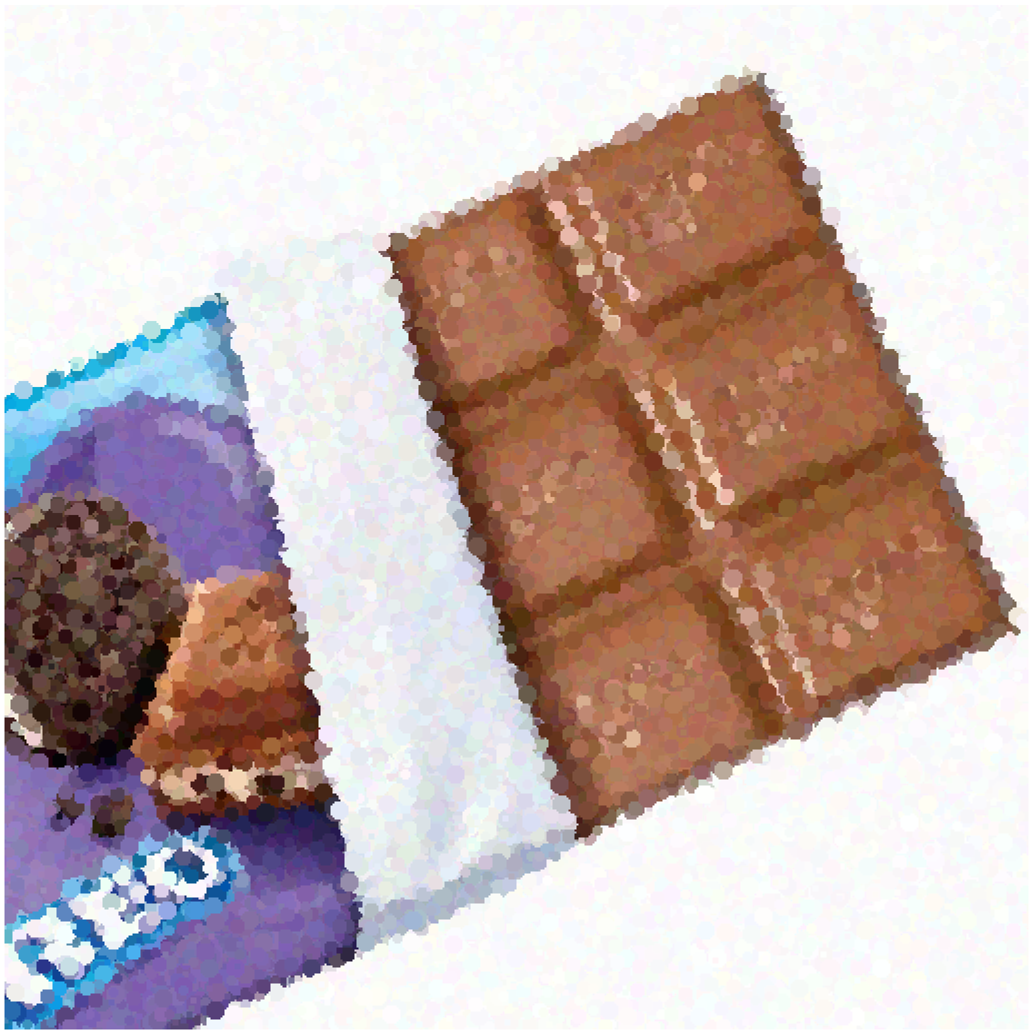
Output: Radius - 4, Quality- 97%, Time Taken - 59s



3. Input - chocolate.png



Output: Radius - 5, Quality - 97%, Time Taken - 63s



For more pictures check out - *results* folder

Optimization -

I used a special optimization in python that allows us to compile the python code in runtime and saves a lot of time during execution as this code is already compiled and we just reuse it by fetching it from the preexisting unit in the memory.

It was possible to do so using *NUMBA* (https://numba.pydata.org/)

Conclusion -

Art is relative and everyone has their own perspective to define it.

Optimization makes the algorithm faster, the maximum time needed for my execution is 1.5 min on some pictures with extreme details.