EVOLUTIONARY COMPUTATION FINAL PROJECT

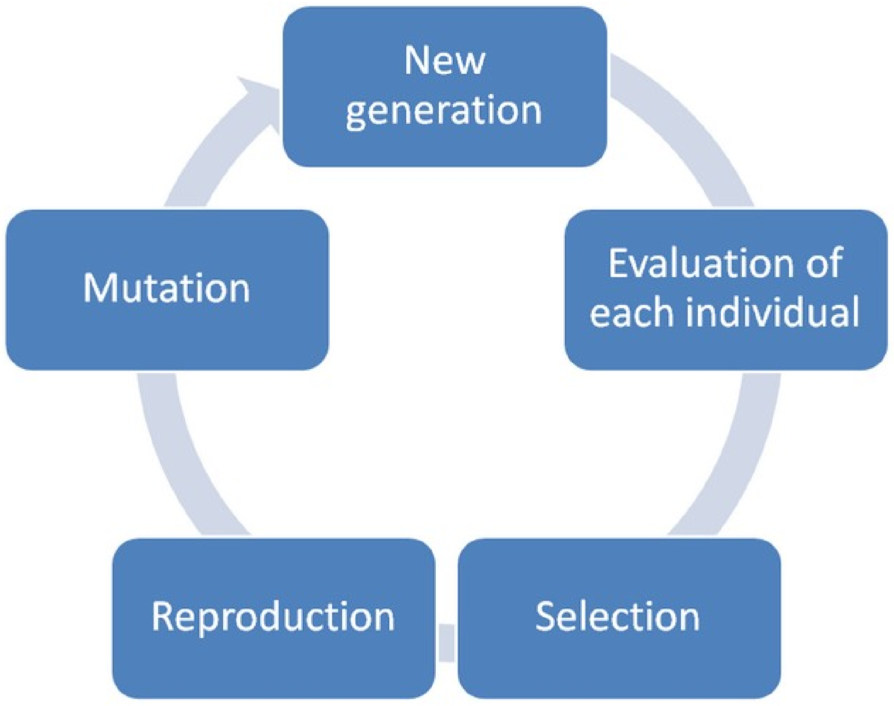
Genetic Algorithm for Reproducing Images is a **Java** project that uses the genetic algorithm. The algorithm reproduces a single image using Genetic Algorithm (GA) by evolving pixel values. This project works with both color and gray images.

Genetic Algorithm

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation



1. **Initial population**

Manually, there is 150 individual in the population.(You can change this value in line 25,in class Main). As initial, 150 individual is generated randomly.

2. **Fitness function**

The fitness function is a little bit more complicated. First, We all know that there is so many square(pixel) in the individual(photo). Every pixel value is converted into **RGB** value for both individual and original image to make calculation easier. And calculated the value difference (distance) between original image and an individual for each pixel. Finally, the value difference is divided by pixel count. Which is our **fitness value**. The smaller value it has, the more fitness value for it. In short, this is **minimization problem.** But the fittest value you will see in the console is go up between 0-1. Because the difference value is **substracted by 1.**

3. **Selection**

First, array of fitness values is sorted by ascending order. Then, the best **n** elements are selected(n is calculated by **selectionRate**). Finally, The random 2 images are selected to crossover from the population of length **n** elements.

4. **Crossover**

One point crossover method is used.

5. **Mutation**

First, there is a random number returned that will decide whether mutation is applied or not. If returned number is smaller than mutation rate, there will be mutate for that gene. That shape item(stack of pixels) is removed from image. Then, that shape will be randomized(changes its parameter like width, height, color), finally new shape is put instead of old one.

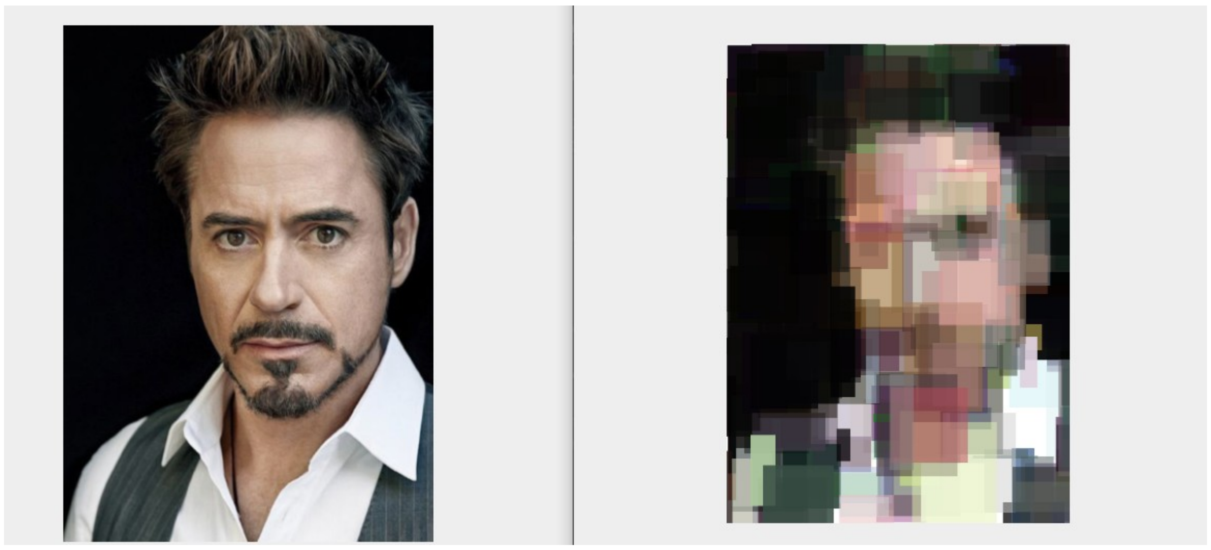


Figure 1

In the Figure 1., it is illustrated the result of reproducing image in 1000 generation.(This process takes 5 minutes)

Implementation Details

- **Genetic** (Class)
    - Genetic algorithm implementation.
- **ImageProcessor** (Class)
    - Utility class for image copy, resize, load, compare operations.
    - This class consist of this operations: copy, resize, load and comparison.
- **RandomGenerator** (Class)
- This class is helper class that helps the genetic algorithm to generate random values like color and other primitive values.
- **Screen** (Class)
    - This is GUI implementation to show original image that you select and the generated image being the fittest one for that generation.
- **TheImage**(Class)
    - Picture class which holds paintable shape in itself. Paintable shape is square which includes more than one pixel to reduce time complexity.
- **Square** (Class)
    - Some shapes that implements `Paintable` interface.

RESULT

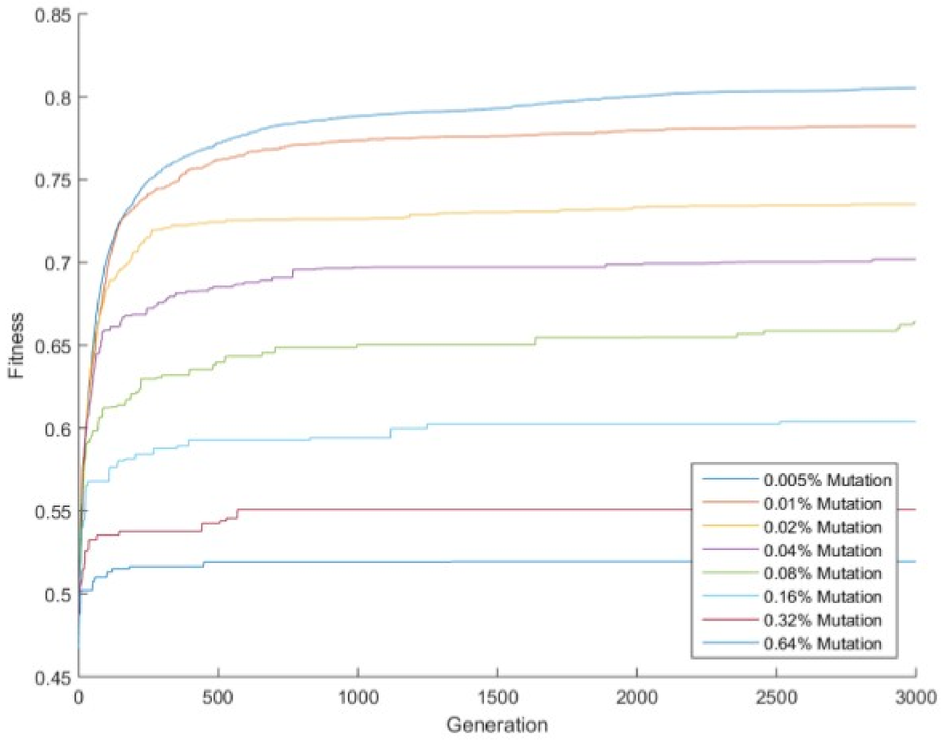


FIGURE 2

As you can see in the figure 3, the fitness value is steadily increasing as mutation rate decreasing. On the other hand, fitness value is approximately is stable after 1000 generation.

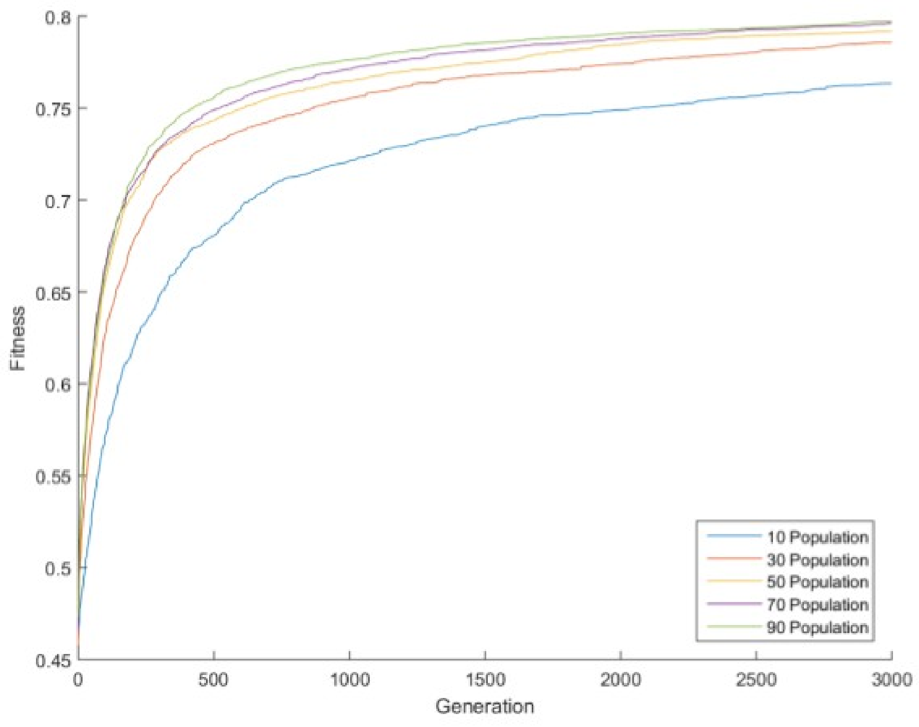


FIGURE 3

This project In this photo (In figure 3) shows us that the more population we have, the more fitness value we reach.

Conclusion

This project can be used to pixelate people who is guilty, who do not want to show himself to camera in the news. Because, the image is pixelated at the end of 5000 generation. Furthermore, it can be used to avoid copyright for that image to use in your website page, blog etc.

WHAT IS THE MAIN PROBLEM WHILE IMPLEMENTING THIS ALGORITHM

While investigating objective function to find fitness value of any individual , I had to struggle with pixel size in an image array. To cope with this problem (because of time complexity), I have reduced square size of image so I created a method named resize to resize the image into width 640, height 640 but this is not good enough to reduce time complexity for generating new generation. Because 640x640 square to compare two images. The second step is to define 300(It can be changed) square to draw image. Every square consists of cluster of pixels.

HOW TO WORK

The parameters are:

```
tring imageFile = "src\\images\\elonmusk.jpg";
int sizeOfPopulation = 150;         // amount of population
int numberOfSquare = 300;           // amount of square
int generation = 5000;
double selectionRate = 0.30;
double mutuationRate = 0.005;
```

You can change variable `imageFile` **to reproduce another photo.**

Variable `sizeOfPopulation` **denotes the number of population.**

Variable `numberOfSquare` **denotes the number of square in the generated image to reduce time complexity.**

Variable `selectionRate` **denotes the rate of selection to select parent.**