Introduction to Artificial intelligence Assignment 2. Evolutionary Algorithm

Innopolis University, Spring semester

Lev Svalov
BS-2018, group 5.

April 2020

Contents

1 Introduction

Hello!
There is a brief intro to the report of the assignment.
I wanted to mention some points and acknowledges about the language decision. With believing that studying is always exploring new things, I have implemented the assignment on C# in which I have not had any experience yet. That is why I need to acknowledge the C# tutorial about the topic. I have followed the tutorial's code through my implementation, thus I needed to make a reference to it.
Also, in the report I didn't put the code itself as I think you can see it in the attached files. Here, I only describe the code and the implementation of algorithm. I think it is more sensible than copy-pasting a lot of lines of project's code into the report. The code is documented and for simpler setting up of the assignment on your local machine, you can just clone it from the github repository.

2 Description of the implementation

The algorithm has basic for evolutionary algorithms structure:

- Create a randomized population made up of chromosomes.
  In particular, create a random population of rectangles with random color.
- **Until "Done" condition**:
  "Done" is determined by 2 conditions:
  - either the best fitness score is reached the predefined value that indicates the completed solution.
  - or the number of the generations is reached the predefined maximum possible value.
    - 1. Calculate fitness score for each member of population and sort the population with respect to it in non-increasing order.
    - 2. Select members using the Crossover and Mutation.
    - 3. Keep some members from the previous population using the elitism value.
- Return the completed output image.

2.1 Representation of the algorithm and image manipulation techniques

I have decided to follow kind of discrete strategy of creating the image and the idea is:
- Image is divided by K rectangles, m*n pixels each.
- Each rectangle has info about its pivot point *(left-upper point)*, the colour of the rectangle, and the lengths of width and height.

In assignment output cases, in order to make the process of generating images less time-consuming, I have decided to make K=64 squares, with length=width=64 pixels each.
As a gene of the my algorithm I consider the square with that contain information about (1) the coordinates of the pivot point *(left-upper point)*, (2) width and height of the rectangle, (3) the color of the rectangle.
And the DNA is the array of such genes.

Simply speaking, I am making the grid(8x8 squares) where each square initially has random colour. And in this case, pivot points of squares are not randomised, because there are 64 rectangles on the image that are represented as a array, the index of the array shows the rectangle position on the image. There are 8 rectangles in each row, we go from left-up rectangle, to right-bottom rectangle. And, therefore, coordinates of the pivot point of each square are calculated using this index and the length and width of the square.

2.2 Selection mechanism

The process of creating new generation is implemented in method **NewGeneration()** in the file **GeneticAlgorithm.cs**.
The main idea is the following: Firstly, we sort the parents in decreasing order of fitness values. In the beginning the program we set up constant number **n** as **elitism**, that indicates that first best(as it's sorted) **n** parent-dna is added to new population without the crossover and mutation. After that, we generate child DNAs using crossover and mutation.

2.3 Fitness function

The process of calculating the fitness value for the particular intermediate image(dna) is implemented in function **FitnessFunction** and it has the following idea: As an argument of this function we can index **i** of the dna in the list of the population. After that, for each rectangle on the image, it runs pixel-by-pixel from left-upper one to right-down one and compares colors with target input image color on this pixel using function **isColorsSimiliar**. This function determines are the colors similar with respect to some tolerance*(that is setting up manually beforehand)*. Thus, it can be not the exactly the same color, but very similar and we will increment fitness score for this pixel. The code of these methods you can see in file **MainProgram.cs**.

2.4 Crossover function

The crossover is done between 2 images(parent DNAs) that have the highest fitness scores in the population, it takes square from 1st parent if random double is less than 0.5, else it takes from the 2nd parent. The implementation of the method in the file **dna.cs**

2.5 Mutation function

The mutation is generating new color for the particular rectangle if the random number is less than the constant mutation rate.
The method **GetRandomGene(int index, boolean ForMutation, MyRectangle gene)** is used for genes' initialization and for mutation. The index is needed here to determine the coordinates on the image of the pivot point of the square.
In particular, when we are using it for mutation(boolean variable = **true**), we randomly setting up only 1 color field(other 2 remain the same), but not all as it is done during the initializing of the gene. It is done with the purpose to make the algorithm less time-consuming.

The implementation itself you can see in attached code, in file **MainProgram.cs**

3 Result overview

3.1 My perception of art

From my perspective, I state that any product that was generated/made without the strict correspondence between source input information and the produced result should be considered as an art, because there exists a kind of space for creativity for creator(whoever and whatever it can be, painter or painting algorithm).
Also, I suppose it is still an art even if this space is evoked by the simple randomness of operations in process of creating the product, since there is no exacting list of instructions that are necessary to do.

3.2 Why output images should be considered as art

I have 2 reasons to consider the produced images as art.

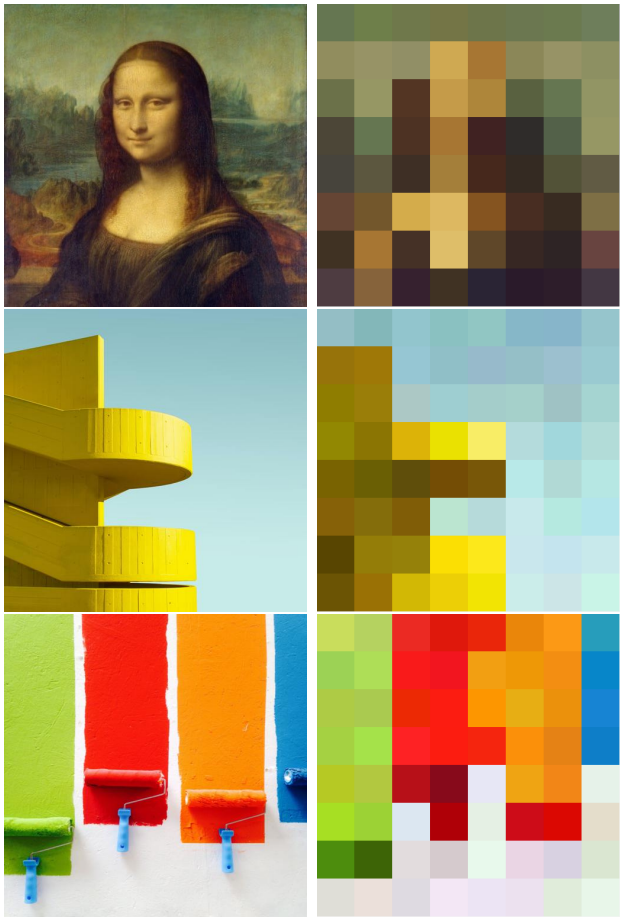
- **Historical argument**. If we refer to the definitions of the art, we will find out that this question is debated for centuries among philosophers. In general, the art is fallen in 3 categories: **representation, expression, and form.**
  I will justify why the produced images are art by matching its characteristics to some of these 3 classes.
  - **Art as Representation or Mimesis.** Plato first developed the idea of art as "mimesis," which, in Greek, means **copying or imitation**. For this reason, the primary meaning of art is defined as the representation or replication of the any product of the world being. If we come back to the produced images, we will see that the output meets the definition of art as a replication, since it produces another new representation of the different things. So that, in sense of representation, it is art.

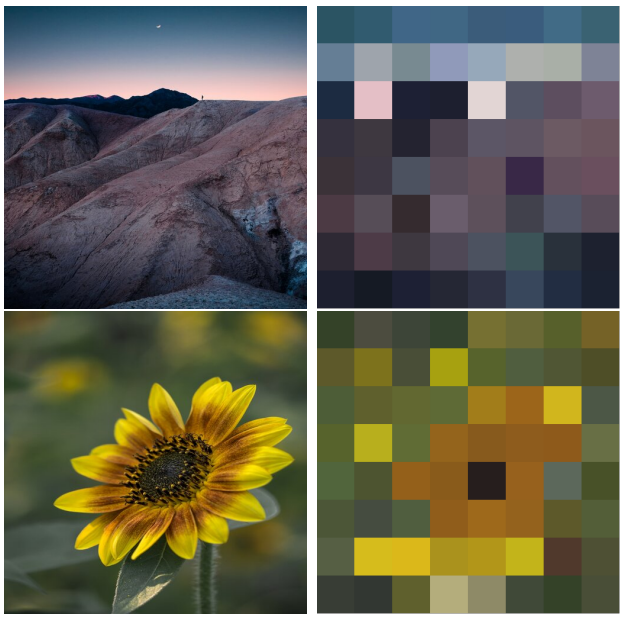
– **Art as Form.** This category was proposed by famous philosopher Immanuel Kant. He believed that art should not have a concept but should be judged only on its formal qualities because the content of a work of art is not of aesthetic interest. Formal qualities became particularly important when art became more abstract in the 20th century. Speaking about the produced images, it has the strict formal quality and form - it has discrete structure, and thus, it gives sort of abstract view of the given source image. That is why, it can be considered as art from this perspective.

The idea is based on the information from the article *Ways of Defining Art*

- **Intuitive argument.** Speaking about the produced images and my algorithm, it is an art because I can not surely say how exactly output will look the for a given input. They will be similar, but there will be **"spaces"** where the algorithm decides the representation by itself*(with random)*. And this **"spaces"** can be different from one try to another. That is why it is an art from my point of view.

3.3 Examples





4 Conclusion

I think there is a large space of improving the representation of the produced output images, since, as you can see from the 2 last images, when source image is becoming more complicated, with complex structure, the output is not close to the origin image. So, the idea of representation can be improved, and I think it will, but not in terms of this assignment.*(Somewhen during the holidays)*.
All in all, I have enjoyed doing the assignment and learning something new for me.