Portfolio Element One: Application of Genetic Algorithms to Generating Paintings

Temur Kholmatov, Innopolis University

October 2020

1 Introduction

Genetic algorithms are frequently applied to optimization tasks because they can discover optimal arrangements without fundamental knowledge of decisions done. These algorithms can find optimal solutions of object features that can be evaluated and weighted based on how good or bad they fit some metric. This project studies the application of genetic algorithms for generating paintings based on images. Each painting is drawn with a set of strokes, and an evolution process is used to explore and exploit the field of possible paintings.

The source code of the project with description is available here: `https://github.com/temur-kh/generated-painting`.

2 Method

This implementation is based on a paper by Nasen and Lewis (2018). However, modifications were done to the original approach due to the fact that some operations like crossover and mutation were explained only generally and no input parameters were provided. That is why new ideas of how to construct those operations were realized, and additional techniques like the aging algorithm were used.

The structure of the program, from a high level, is as follows:

1. The generator function generates n initial random paintings.
2. The selector function selects $\frac{n}{3}$ paintings with the highest score.
3. The crossover function pairs selected paintings by two and combines strokes from each pair and generate new paintings. Those paintings are added to the population.
4. The selector function selects a specified number of paintings and with a specified probability mutates randomly chosen parameter of strokes.
5. The aging algorithm increases ages and remove paintings older than specified age. This technique helps increasing variation of paintings in the population.
6. The selector function selects top n paintings for the population for the next epoch.

7. The algorithm starts a new epoch from the step 2.

The fitness function is taken from the original paper and is just the maximal possible score minus a mean absolute difference between images.

2.1 Painting generator

The painting generator is used on the first stage when the initial population is to be created. The population consists of the n paintings where n. Each painting is a combination of m strokes with 5 parameters: color (contains 3 values in RGB color model), length, width, position (x and y values), and rotation degree.

According to the original paper, the parameter space of the problem is too large. That is why starting with truly random initialization of strokes puts the model far away from the optimal solution. As a result, the genetic algorithm spends significantly more iterations to converge to a desirable painting. Instead, the algorithm uses an "informed" random generator. Here is the list of modifications and constraints applied:

- Positions of strokes are still generated randomly on an image plot.
- The initial color is set to be the color of the original image at the chosen stroke position.
- The length and width parameters are limited with lower and upper bound.
- All parameters are chosen randomly according to their constraints.

This approach help to greatly increase the starting fitness scores of paintings.

2.2 Selection

The selection function is used mainly in three parts: before the crossover operation, before the mutation operation and at the end of each epoch to reduce the size of the population to n. There are two modes of selection:

- Random selection
- Selection by the maximum fitness score.

2.3 Crossover

As suggested in the paper, we select $\frac{n}{3}$ of the population for the crossover operation. However, in the current implementation, the selection is done in the maximum-score mode compared to the random selection proposed in the paper. For the crossover, the authors suggest taking half of the strokes from one parent painting and half of the strokes from the other and combining them. However, they do not explain explicitly the process of stroke selection. Here is the interpretation of the stroke selection and the overall crossover process we are to follow:

- Divide the selection into pairs of paintings that will be parents.
- Sort the strokes of each painting by position.

- Randomly choose strokes from both paintings so that each parent contributes the same number of strokes to the child painting.

The sorting operation helps to organize strokes in such a way that a child painting will have almost uniformly distributed strokes.

2.4 Mutation

After the children are added to the population, we select p random paintings for the mutation operation (p is a user-defined parameter). There are such user-defined parameters like the probability of selecting a stroke for a mutation. Additionally, each stroke attribute has a parameter that defines the maximum possible step when changing some value. The overall process of mutation can be described like this:

- For each painting select strokes to be mutated with the user-defined probability.
- For each stroke randomly select one attribute to be mutated.
- For each attribute select a random step with respect to the constraints and the user-defined parameters of maximum step values.
- Add the step value to the original value of an attribute.

2.5 Aging algorithm

The aging algorithm is the process of tracking the lifetime of each painting in the population and removing those who lived for more than a epochs. This technique is helpful as it benefits to exploration. The process is quite simple:

- At the end of each epoch, remove paintings with an age larger than a, where a is a user-defined parameter.
- Afterward, increase the age for every painting.

2.6 Picture rendering

Finally, we render pictures before the process of fitness evaluation. This is done with the help of the OpenCV library. The algorithm takes a purely white or black matrix of pixels and draws all strokes in random order.

3 Evaluation and Results

3.1 Fitness function

The fitness function is the most important part of a genetic algorithm because it allows evaluating whether the process goes in the right direction, and the solution converges to an optimal one. Since we generate a painting from a source picture, we can use it for the distance-based fitness score. The authors of the above-mentioned paper suggest using simple *L1 distance* between the RGB values of each pixel of a generated painting and the corresponding pixels of the original image. This operation stays the same for our implementation.

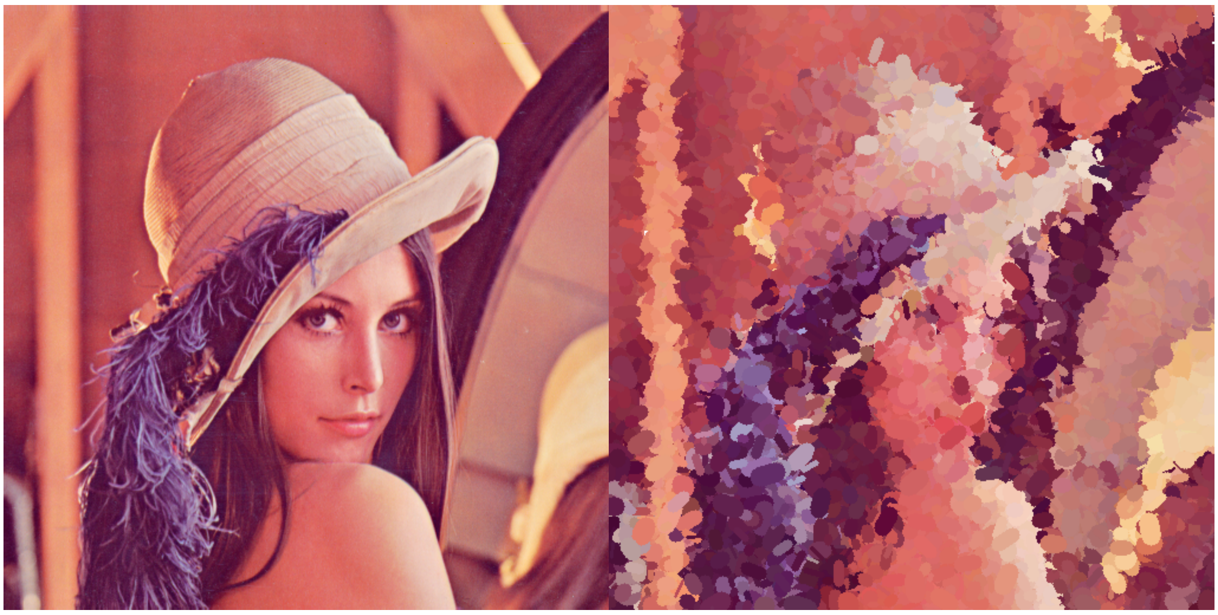

Figure 1: The original picture of Lenna on the left and a generated painting on the right.

3.2 Results

In this section, several examples of generated paintings are provided. In Figure 1, you may see Lenna and the corresponding generated painting, whereas, in Figure 2, there is an example of painting generation for a picture of nature. In both cases the application was run with the default parameters: number of epochs = 25, population size = 50, number of mutations per epoch = 25, stroke mutation probability = 0.2, number of strokes per painting = 20000, and etc.

4 Further Work

One may improve the results of the painting generator by fine-tuning its user-defined parameters. Also, one may use other techniques like aging algorithm to balance between exploration and exploitation processes.

References

[1] Hansen, A., Lewis, M. C. (2018). Applying Genetic Algorithms to Generating Paintings. In Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV) (pp. 92-98). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

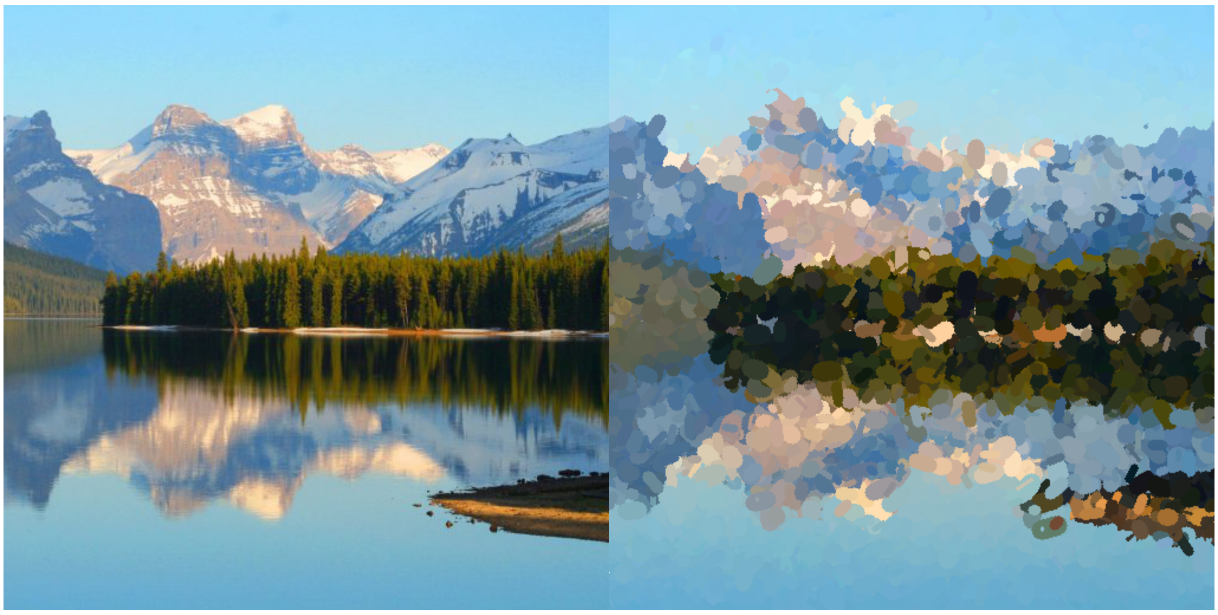

Figure 2: The original picture of nature on the left and a generated painting on the right.