Assignment II - "Computers can do art!"

Roman Nabiullin

1 What is art for you?

Needs to say that there is no straightforward answer for this debatable question. As for me, art is anything created to express oneself, tell a story, show something beautiful to the world, inspire people, cause a reflection. Humanity collected such pieces of art for many millennia, and nowadays one may find them in museums, exhibitions, or the internet. However, art is not only music written by a genius composer or an ancient sculpture. It can be a hand-drawing for a kid and their parents or street graffiti for teenagers. Why not? Art is subjective. It can be even RGB images generated by the computer. Although those pictures are created by a machine, they are charming. I definitely will use a couple of them as a desktop wallpaper. If such computer-generated images inspire me, I shall call them art. That is my personal opinion.

2 Image processing, CUDA and optimization

An image of .jpg or .png format is opened and converted into a NumPy array.

The size of this array is

$$\text{IMAGE\_HEIGHT} \times \text{IMAGE\_WIDTH} \times 3$$

where 3 stands for RGB channels.

It was decided to use python library numba that provides just-in-time compilation and CUDA facility. This allows to increase the performance dramatically - reduce the execution time from several hours per standard epoch (epoch is described in the below sections) to $3-6$ minutes per standard epoch.

Those CUDA features were used: GPU reduction for efficiently calculating the fitness value, parallel computing for performing mutations.

Reference links:

- Just-in-time compilation
- numpy and numba libraries
- NVidia CUDA technology
- GPU reduction

3 Algorithm description

```
function run_evolution(inital_popualtion, epoch_duration, epoch_count)
{
   population <- initial_population;
   for epoch_id from 1 to epoch_count
   {
      for generation_id from 1 to epoch_duration
      {
         population <- run_generation(population);
      }
      save epoch snapshot;
   }

   return the best individual from population;
}
```

Figure 1: Evolution function

```
function run_generation(current_generation)
{
   next_generation <- empty list;
   best_individual <- individual from current_generation with the best fitness;
   add best_individual to next_generation;
   repeat
   {
      successor <- selection(current_generation);
      successor <- mutation(successor);
      add successor to next_generation;
   } until next_generation contatins exactly population_size individuals;

   return next_generation;
}
```

Figure 2: Generation function

It was decided to use the above evolutionary algorithm. It includes selection and mutation processes that are explained later. For the sake of readability, the algorithm is split into two functions.

Since in most cases, it is almost impossible to observe any difference after a single mutation, we introduce a concept of an epoch.

Epoch - is a batch of, for instance, 10k generations (as in standard epoch). Snapshots including best-fit chromosomes and best-fitnesses are taken and saved after each epoch.

As for the number of epochs, it depends on the source image size. It is enough to have 10 standard epochs for a 512x512 picture. So that there will be approximately 100k generations.

4 Chromosome representation

In our case, a gene is a triangle of some shape and color. A chromosome is represented as an ordered sequence of genes.

It was decided to maintain a chromosome as a composed NumPy array for the sake of algorithm efficiency.

5 Fitness function

Given two images of the same size H × W × 3. The fitness value - distance between these two images is going to be the following double sum

$$\sum_{i=1}^{H}\sum_{j=1}^{W}\frac{\sqrt{\Delta R_{ij}^2+\Delta G_{ij}^2+\Delta B_{ij}^2}}{\sqrt{255^2+255^2+255^2}}$$

where $\Delta R_{ij}, \Delta G_{ij}, \Delta B_{ij}$ are difference for each of RGB channel for i, j pixel.

Basically, the fitness value here is the sum of ratios of Euclidean distance between 2 pixels and the maximal Euclidean distance between 2 pixels. The range for fitness values $[0 \dots 3 \cdot H \cdot W]$.

So, if the fitness value is small then two images are rather similar. If fitness value is big, two images are completely different.

As for the fitness limit, we can set it to

$$0.08 \cdot \text{max\_fitness} = 0.24HW$$

Tests demonstrate that 8% of total difference is enough for a human (at least me) to call two images pretty similar.

6 Selection

The roulette wheel selection method has been chosen. As for the probabilities, we use the following approach:

If fitness values are $f = [f_1, \dots f_n]$ then respective probabilities are

$$p = \left[\frac{f_1^{-1}}{\sum f^{-1}}, \dots \frac{f_n^{-1}}{\sum f^{-1}}\right] \text{ where } f^{-1} = [f_1^{-1}, \dots f_n^{-1}]$$

We should use inverted fitness values for calculations due to the nature of our fitness function (section 2.3).

7 Crossover

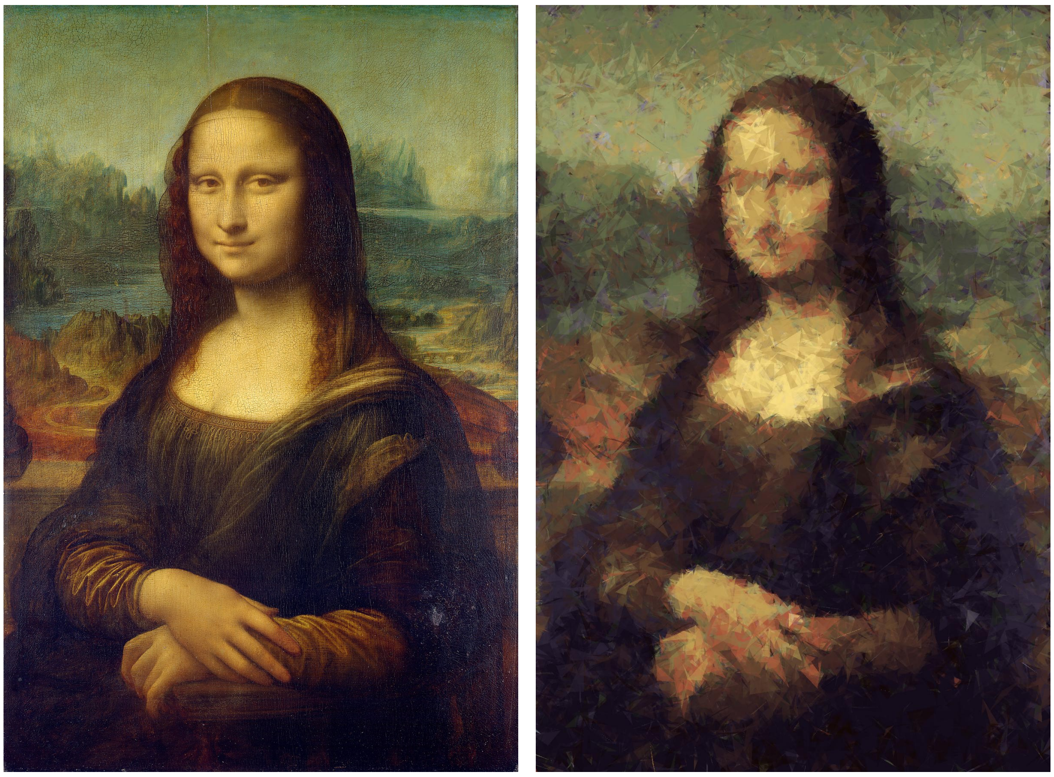
It was decided not to use any crossover function.

8 Mutation

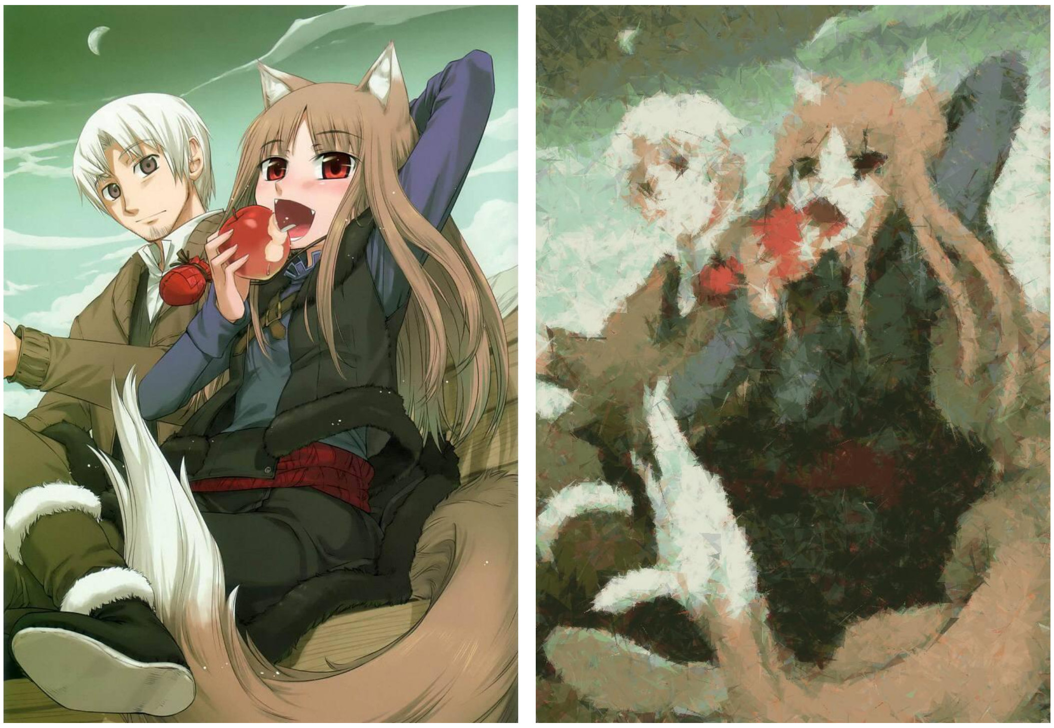
In our case, a mutation is a process of adding extra gene(s) to the tail of the chromosome. As for the implementation:

1. A candidate image is taken.
2. New overlay with a triangle is created.
3. Overlay is merged with the candidate image using alpha-blending(link).

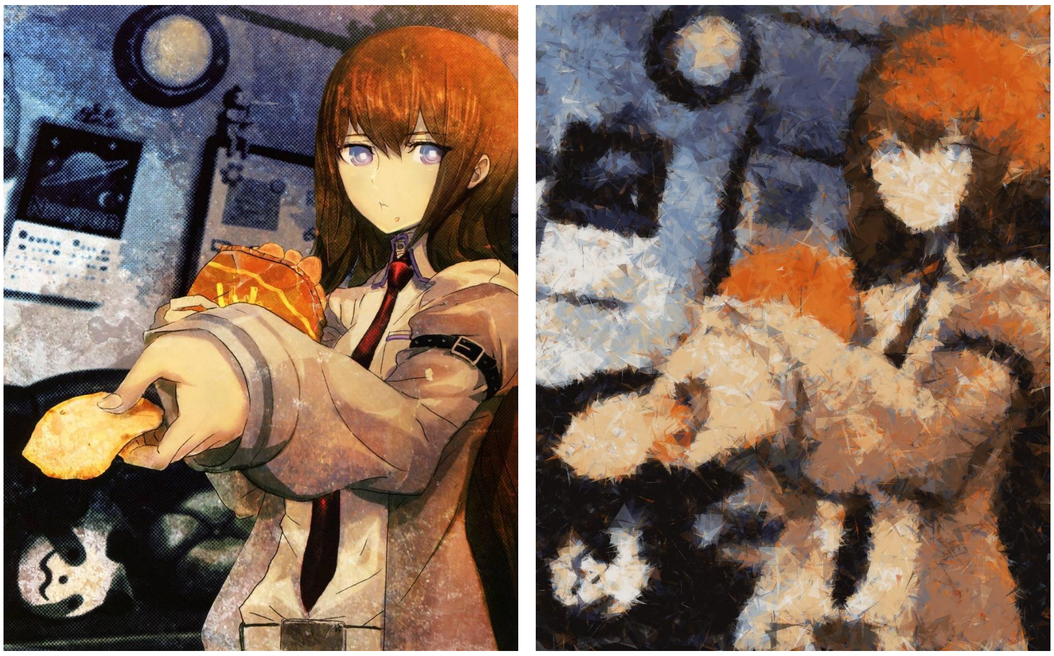
9 Generated images

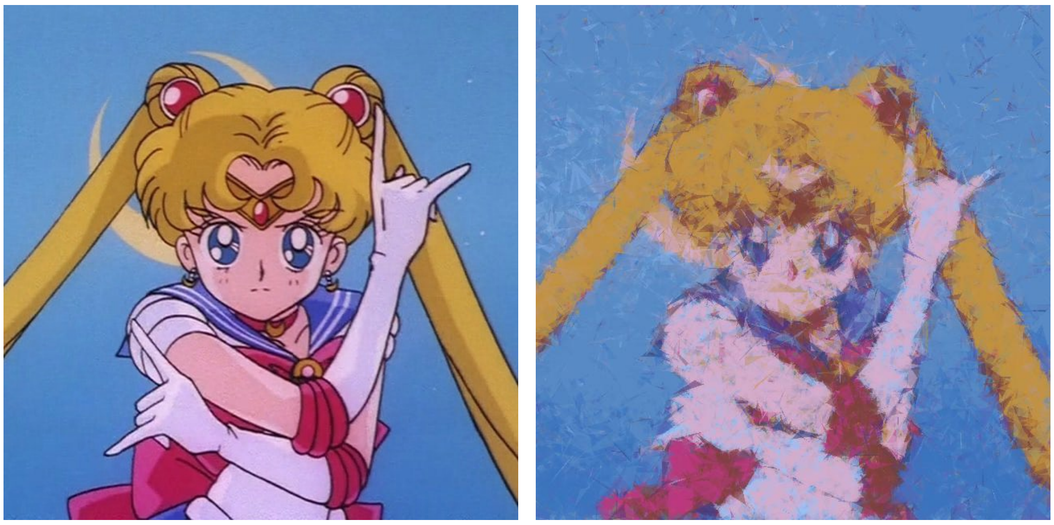


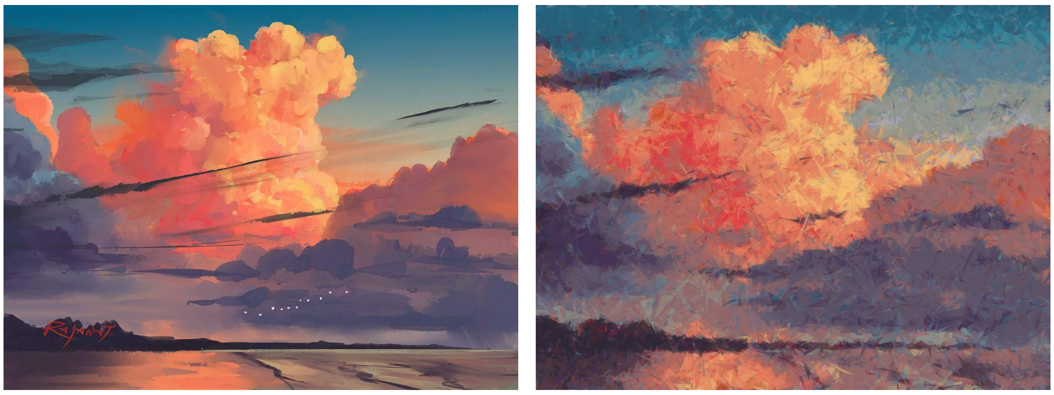
Da Vinci, L. (1503-1519). Mona Lisa [Painting].



文倉十 (Jū Ayakura). 狼と香辛料 (Ōkami to Kōshinryō) - Manga volume I cover (2008).


Huke, (2013). Steins;gate Art Works imaginations of huke [Painting].


Takeuchi et al., (1992-1997). Sailor Moon [TV series].


Rajawat, S. (2021). Daily Sketches [Painting].